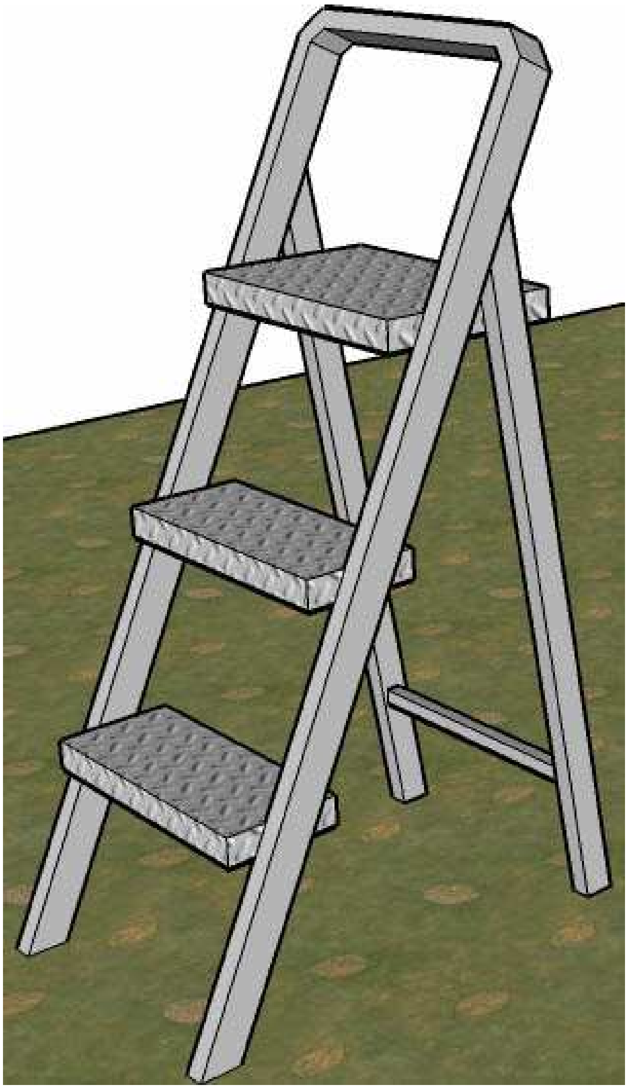
ترجمة كتاب

فيجوال بيسك ٢٠٠٨ خطوة خطوة

المؤلف: Michael Halvorson



صفحة الكتاب:

http://www.amazon.com/Microsoft-Visual-Basic-2008-Step/dp/0735625379/

ترجمة: مروان المفلحي

الموقع : فيجوال بيسك للعرب: http://vb4arab.com/vb/

المدونة: http://marwanvb.blogspot.com/

البريد: vbdotnet2008sbs@gmail.com

هل هذا الكتاب قانوني

ملاحظة: لابد من شراء هذا الكتاب النسخة الإنجليزية من صفحة المؤلف على الأمازون، لكي يكون هذا الكتاب قانوني، الترجمة فقط إلى العربية مجاناً، لكن الكتاب نفسه يجب شراؤه من الأمازون. بعبارة أُخرى يجب شراء النسخة الإنجليزية من المؤلف، لابد من توضيح هذا الأمر لجميع القراء حتى لا تضيع الحقوق الفكرية للمؤلف.

هذه الصفحة تُركت بيضاء بشكل مقصود

السلام عليكم ورحمة الله وبركاته، إلى أعضاء منتدى فيجوال بيسك للعرب وإلى الطلاب العرب المحترمون. أحببت أن أضع بين يديكم ترجمة متواضعة لكتاب

Visual basic .NET 2008 Step by Step، فيجوال بيسك دوت نت 2008 خطوة خطوة.

هذا الكتاب للمؤلف Michael Halvorson الذي ألف وشارك في تأليف أكثر من ثلاثين كتابا في تقنية المعلومات من ضمنها فيجوال بيسك 6 خطوة خطوة وكذلك فيجوال 2005 خطوة خطوة وكتب أخرى عن الأوفس واكس بي وغيرها.

الكتاب يعتبر مقدمة شاملة للبرمجة باستخدام فيجوال بيسك 2008 بحسب كلام مؤلفه ويحتوى على أربعة أجزاء تشمل عشرون فصلاً وواحد وستون تمرين برمجي مبسط ومفصل لعمل العديد من التطبيقات البرمجية والتي تتوافق مع ويندوز والعديد من متصفحات الانترنت.

سأحاول أن أكون مترجما مع قليلاً من التصرف حتى تتوافق الترجمة مع طريقة فهمنا وسأحاول أن أحافظ على ترتيب الأبواب والفصول بحسب ترتيب المؤلف.

ملاحظات حول عملية الترجمة:

١- استخدم بعض الأحيان مصطلح الفيجوال بيسك 2008 وبعض الأحيان مصطلح الفيجوال 2008 ولتوضيح اللبس الذي قد يحث عند بعض الإخوة من أن المصطلحين عبارة عن مترادفات، أقول لهم الفيجوال 2008 هو عبارة عن بيئة التطوير التي تحتوي على السي شارب والفيجوال بيسك والسي بلس بلس وغيرها من لغات البرمجة أما الفيجوال بيسك 2008 فهو لغة البرمجة فيجوال بيسك 2008 فقط.

٢- للعديد من الأسباب أتكلم في الكتاب باسم المؤلف الأصلي للكتاب لترجمة بعض الجمل في الكتاب وفي البعض الأخر أتكلم باسمي (المترجم للكتاب). وستجد بعض العبارات بطريقة المُخاطب وبعض العبارات كأني أُحادث نفسي وبقية القراء.

٣- تم إهمال بعض الأمور الغير مهمة عند الترجمة وتم تكثيف الترجمة عند الجوانب الهامة وإضافة ما يمكن إضافته للترجمة.

٤- ليس لدي الخبرة القوية بالفيجوال دوت نت ولكني رأيت انه من خلال ترجمة مثل هذا الكتاب قد ارفع من قدراتي في البرمجة وأتعلمها بطريقة أفضل لذلك فاعذروني في حالة وجود بعض الأخطاء. وراسلوني لتعديلها في الكتاب.

٥- بعض المصطلحات أقوم بنشر المصطلح الأصلي إذا كانت الترجمة ركيكة أو غير متأكد منها.

٦- يجب على القارئ القراءة من أكثر من مصدر وعدم الإعتماد على مصدر واحد في المطالعة، فهذا الكتاب بالرغم من المعلومات الكثيرة المتوفرة فيه، لكن لابد من الإطلاع على المواقع التعليمية والكتب المتخصصة.

ونبدأ إن شاء الله مع الكتاب:

ما هو فيجوال بيسك 2008

فيجوال بيسك 2008 هو أداة تطويرية تستخدم لإنتاج التطبيقات والبرامج ويحتوي على العديد من الإعدادات والأكواد الجاهزة التي تيسر علينا كتابة الكود وتصميم البرامج التي تعمل تحت منصة الويندوز وكذلك البرامج التي تعمل على الانترنت والأجهزة الكفية (أجهزة الجيب). فيجوال بيسك يساعد على زيادة الإنتاجية عند تصميم البرامج خاصة البرامج المتعلقة بقواعد البيانات وبرامج

الانترنت. وهنا ملاحظة أنه عندما تعتاد على استخدام بيئة التطوير الخاصة بالفيجوال 2008 فانك ستصبح قادرا على استخدام نفس الأدوات التي تستخدمها في فيجوال بيسك تستطيع أن تستخدمها مع فيجوال سي شارب 2008 وكذلك سي بلس بلس 2008 وغيرها من أدوات التطوير المضمنة في Visual Studio 2008.

نسخ الفيجوال بيسك دوت نت

بدأت الفيجوال دوت نت في فبراير 2002 ثم نسخة فيجوال دوت نت 2003 في مارس 2003 ثم الفيجوال دوت نت 2005 في أواخر 2005 . وبعد العديد من التطويرات التي تمت على لغة الفيجوال دوت نت 2005 تم إصدار الفيجوال دوت نت 2008 في بداية عام 2008. ويعتبر فيجوال بيسك دوت نت 2008 مربوطاً ربطاً وثيقاً مع الفيجوال المرئي 2008 والذي يشمل على السي شارب 2008 والسي بلس بلس 2008 والعديد من لغات التطوير.

توجد العديد من نسخ الفيجوال استوديو 2008 وهيٍ Standard Edition و Professional Edition و Team Suite و Express Edition. هذا الكتاب يتوافق مع جميع النسخ المذكورة أعلاه. وبالرغم من التشابه بين فيجوال بيسك 2005 وفيجوال بيسك 2008 إلا أن هناك فروق هامة وجوهرية بين نسخة الـــ 2005 ونسخة الـــ 2008 لذلك فيفضل أن تملك نسخة الفيجوال بيسك 2008 لتواصل مع هذا الكتاب.

الانتقال من فيجوال بيسك 6

كانت فيجوال بيسك 6 هي النسخة التي سبقت فيجوال دوت نت وأصدرت (الفيجوال 6) قبل عشر سنوات تقريباً في سبتمبر 1998. لا يزال هناك العديد من المبرمجين الذين يعملون تطبيقاتهم على هذه النسخة القديمة ولكنني ألومهم على تعلقهم الشديد بهذه النسخة لان تطوير تطبيق معقد أو تطبيق احترافي بهذه اللغة شيئا صعب نوعا ما.

باستخدام نسخة الفيجوال دوت نت 2008 بإستطاعت المبرمجين تصميم العديد من التطبيقات الاحترافية التي تعمل تحت نظام الويندوز وكذلك تصميم مواقع الانترنت بسهولة أكثر من استخدام الفيجوال 6، يستطيعون من تصميم التطبيقات التي تنافس التطبيقات المصممة بالسي شارب وبالسي بلس سلس وتطبيقات الجافا. وكذلك تعتبر لغة الفيجوال بيسك 2008 أسهل من بقية أدوات التطوير الأخرى المضمنة في فيجوال استوديو 2008 (على الأقل من وجهة نظر المؤلف والمترجم).

هناك Wizard يقوم بتطوير المشاريع المصممة بالفيجوال 6 إلى فيجوال بيسك 2008. طبعا هذا الـــ Wizard ليس دقيقا ولا يعمل مع كل المشاريع المصممة بالفيجوال بيسك 6 ولكنه مثالي نوعا ما وسوف تعلم (يا مستخدم فيجوال بيسك 6) أن العديد من الأوامر البرمجية والطرق والدوال والتصريحات والخصائص التي كنت تستخدمها في فيجوال 6 لا زالت موجودة في فيجوال بيسك 2008. في هذا الكتاب سأقوم بتقديم العديد من الملاحظات لمبرمجين الفيجوال بيسك 6 للانتقال إلى الدوت نت. لأني سابقا كنت مبرمج فيجوال بيسك 6 واعرف التغييرات التي حدثت في اللغة بين فيجوال بيسك 6 وفيجوال بيسك دوت نت. فلذلك ستعرف خلال قراءة هذا الكتاب العديد من الملاحظات بين الفينة والأخرى سأقوم بذكرها كيف تغيرت الإجراءات المنطقية والدوال وكيف تستخدم معرفتك بالفيجوال 6 لتصبح مبرمجا ماهرا وخبيرا في لغة الفيجوال بيسك دوت نت 2008 (راجع هذا الموضوع للأخ سامر سلو عن ترقية المشاريع من فيجوال 6 إلى دوت نت

http://vb4arab.com/vb/showthread.php?t=9932 وراجع هذا الموضوع للأخ الفجر الأبيض لتطوير كود معين من فيجوال بيسك 6 إلى فيجوال بيسك 2008

http://vb4arab.com/vb/showthread.php?t=16724).

ونصيحة مني لكل المبرمجين أن يقوموا باستخدام كل ما يقع تحت يديهم من معلومات وخبرة برمجية وألا يتعلموا فقط ما كانوا يخططون له فقط فكل ما يقع تحت يديهم يجب أن يتعلموه. فمثلاً

التعامل مع اللوغاريتمات، قواعد البيانات، البرمجة كائنية التوجه، وغيرها من الأمور البرمجية، تعلم مثل هذه الأمور دفعة واحدة يساعد المبرمجين على تصميم التطبيقات الاحترافية والمميزة.

من أين تبدأ في هذا الكتاب

هذا الكتاب تم تصميمه ليساعدك في تطوير معرفتك بالفيجوال دوت نت 2008 في العديد من الاتجاهات فإذا كنت مبرمج فيجوال 6 أو منتقلا من فيجوال بيسك 2005 أو منتقلا حتى من لغة أخرى ستجد في هذا الكتاب ضالتك التي تنشدها ولكن استعين بالجدول التالي لتعرف من أين تبدأ مع هذا الكتاب.

إذا كنت	اتبع التالي
جديد على البرمجة	تعلم المهارات الأساسية في فيجوال 2008 وذلك بدراسة الفصول من 1–17 في هذا الكتاب. قم بالاطلاع على الجزء الرابع "برمجة قواعد البيانات ومواقع الانترنت" إطلاعاً وبحسب رغبتك في المتابعة.
تريد الانتقال من فيجوال 2002، 2003 أو 2005	أقرأ من الفصل الأول حتى الرابع، اطلع على الفصول من 5 إلى 17 إطلاعاً(قراءة بدون تعمق). ثم أكمل قراءة الفصول 18 إلى 20. لتتعرف على بعض التطويرات التي تمت في فيجوال بيسك 2008 أقرأ الفصول: 1، 4، 5، 7، 8، 13، 18، 19، 20
تريد الانتقال من	أقرا الفصول 1 إلى 4 بتمعن لكي تعرف التطويرات التي

فيجوال بيسك 6	حدثت في اللغة وفي بيئة تطوير 2008. تنبه للملاحظات التي سأضعها خلال صفحات هذا الكتاب لمبرمجين الفيجوال 6. أقرا الفصول 5 إلى 13 بسرعة لتتعرف على التغيرات التي حدثت في اللغة مثلا في استخدام المتغيرات وغيرها من الأمور البرمجية وركز على الفصول 5، 6، 9، 12. قم بتطبيق ما تعلمته في الفصول 14 إلى 20 لتتعرف على المزايا الجديد في فيجوال 2008 في واجهة المستخدم، قواعد البيانات وكذلك في تصميم مواقع الانترنت.
بعد قراءة كل فصل ركز على الخلاصة في أخر كل فصل لتتذكر ما تعلمته.	

قبل أن ندخل في الكتاب ونبدأ بالجزء الأول أريد أن أوضح إن هذه الترجمة ما هي إلا جهد شخصي (مليء بالأخطاء)، لنتعاون سوياً من اجل تنقيح الترجمة ومراجعتها وبعون الله تعالى سأقوم بين الفينة والأخرى باعتماد تعديلاتكم وإضافتها إلى الترجمة.

وتقبلوا خالص تحياتي،،،

مروان المفلحي – صنعاء

الجزء الأول: البداية مع فيجوال بيسك 2008

ويحتوي على:

الفصل الأول: التعرف على بيئة التطوير.

الفصل الثاني: كتابة برنامجك الأول.

الفصل الثالث: التعامل مع صناديق الأدوات.

الفصل الرابع: التعامل مع القوائم والأدوات وصناديق الحوار.

الفصل الأول: التعرف على بيئة التطوير.

مع نسخة الكتاب يوجد مجموعة من الأمثلة لكننا سنستغني عنهم وسنعيد كتابتهم معكم من جديد ليتم شرحهم بشكل مفصل كل تمرين في حينه.

افتح الفيجوال بيسك دوت نت 2008

من قائمة إبدأ > كافة البرامج > Microsoft Studio 2008 > اختر Microsoft Studio 2008 إذا كانت هذه هي المرة الأولى لك في فتح البرنامج سوف يتأخر البرنامج لبضع لحظات لتركيب الإعدادات الخاصة، سيطلب منك خلال الفتح اختيار اللغة التي تعمل عليها (فيجوال بيسك أو سي شارب) اختر فيجول بيسك ليختار لك إعدادات بيئة الفيجوال بيسك.

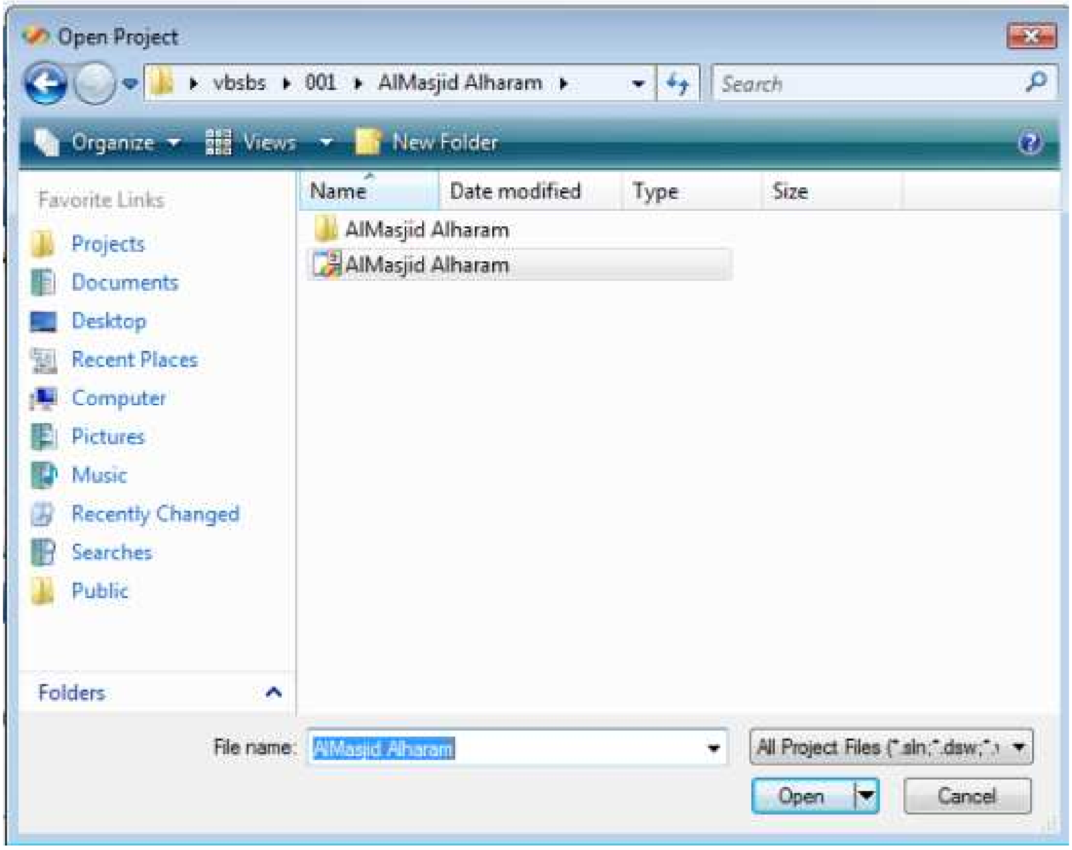
عند فتح البرنامج سوف تشاهد بيئة التطوير وتحتوي على العديد من الأدوات والقوائم والمكونات (تدعى هذه النافذة في ي بعض الأحيان نافذة الأدوات) سوف تشاهد كذلك صفحة البدء وعليها العديد من الوصلات مقالات من الـــ MSDN (مكتبة تعليمات البرنامج) وكذلك خيارات المشروع.

صفحة البدء تحتوى على العديد من المعلومات والمصادر التي تفيدك خلال تصميم تطبيقاتك، وكذلك يوجد مصادر ووصلات لمعلومات على الانترنت تربطك بمجتمع فيجوال بيسك الخارجي.

في البداية افتح مشروح موجود مسبقا في جهازك لذلك خذ أول مشروع في هذا الكتاب في مجلد رقم 001 المرفق مع هذا الكتاب.

لفتح المشروع موجود مسبقا في جهازك استخدم أسلوب الفتح الموجود في البرنامج مثل الذي موجود في وورد Wordأو اكسل Excel وذلك بالضغط على Open Project أو أضغط على

Ctrl+O ، بعد الضغط على Open Project نبحث عن المشروع الذي نريد أن نفتحه وللتجربة مرفق لكم تمرين في مجلد رقم 001 نذهب إليه نبحث عن ملف AlMasjid Alharam انظر الصورة:





بعض الصور المعروضة خلال صفحات هذا الكتاب ملتقطة من ويندوز XP والبعض الأخر من Vista.



إذا كانت إعدادات الويندوز عندك تظهر امتداد الملفات فإن امتداد ملف مشاريع فيجوال بيسك هو sln.

سيقوم بعدها الفيجوال بيسك بتحميل تطبيق المسجد الحرام (الفورم التابع للتطبيق والكود مع إعدادات التطبيق وكذلك خصائص التطبيق) ستلاحظ عند اعلي يمين الشاشة Solution Explorer والذي من خلاله تستطيع اختيار الفورم الذي تريد أن تعمل عليه إذا كان التطبيق

الذي تعمل عليه يحتوي على أكثر من تطبيق وفي تطبيق المسجد الحرام الذي نحن بصدده فانه يوجد فورم واحد فقط نقوم بالضغط عليه مرتين لفتحة ومعرفة والأكواد التي فيه وكذلك خصائصه.

تأكد من انك تستخدم برنامج Visual Studio 2008 لفتح الملفات المرفقة لان النسخ القديمة 2002، 2003، 2005 قد لا تفتح التطبيقات المصممة بالفيجوال 2008

بعد فتح التطبيق تستطيع أن تقوم ببناءة بواسطة الضغط على F5 أو الذهاب إلى Debug في قائمة البرنامج واختيار Start Debugging سيظهر لك التطبيق الذي صممته لك مسبقاً اضغط على زر إظهار الصورة ستظهر لك صورة المسجد الحرام. قم بإغلاق التطبيق أو بالضغط على زر خروج لنعرف أكثر عن هذا التطبيق.

اذهب إلى Solution Explorer واختر Form1 انقر عليها ليظهر لك الفورم الذي يعمل عند تشغيل التطبيق سيظهر لك الفورم ولكن مكتوب أعلى منه [Form1.vb [Design فكلمة Design تعني انك الآن في مرحلة تصميم واجهة المستخدم التي سيتعامل معها مستخدمو برنامجك. هناك شيء هام وهو واجهة الأوامر أو منطقة الكود التي تقوم بكتابة الأوامر فيها أو الأكواد (الشفرات البرمجية). لإظهار منطقة الكود اضغط Right Click على الفورم واختار View Code أو اضغط Double Click على الفورم. عند فتح منطقة الكود ستجد هذا الكود في منطقة الكود والذي قد وضعته لك مسبقاً:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        ' انظر اخي المبتدئ في البرمجة عندما نريد ان نكتب لنا تذكير في منطقة
الكود فإننا نكتب قبله '
        ' لا ان المكتوب باللون الاخضر لا علاقة له بالاوامر البرمجية لكن لغرض
التذكير أو الاغراض التعليمية فقط
        ' يستخدم هذا الكود للخروج من برنامجك
        End
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        ' هذا الكود لاظهار الصورة
        PictureBox1.Visible = True
End Sub
```

لاحظ أن المكتوب باللون الأخضر لا علاقة له بالكود البرمجي وإنما للتوضيح وللتذكير فقط.

سنعود لمناقشة الكود وكل التفاصيل عنه لاحقا.

الفرق بين Project و Solution

إذا تصفحت المجلد الذي ينشأ عند استحداث تطبيق جديد ستجد انه يحتوى على أكثر من ملف (كل فورم له ملف الخ) من ضمن هذه الملفات ستجد ملفين احدهما بالامتداد .sln والأخر بالامتداد .vbproj الأول يعبر عن الـــ Solution (تطبيق) والثاني عن الـــ Project (مشروع)

ستلاحظ أيضا وجود رقم 9 (مصغرة) في أيقونة التطبيق هذا يعني أن هذا التطبيق مصمم الفيجوال 2008 (والتي تعتبر النسخة رقم 9 للفيجوال).

فالمشروع Project يحتوي على العديد من الملفات الخاصة به لتنفيذ مهمة برمجية معينه أما التطبيق Solution فيحتوى على تطبيق واحد أو أكثر من المشاريع المترابطة والمتعلقة ببعضها ببعض. فالمثال الذي أوردته لك هو عبارة عن تطبيق يحتوي على مشروع واحد فقط لذلك فإذا فتحت الملف ذو الامتداد .sln أو الملف ذو الامتداد .vbproj فيفتح لك نفس التطبيق. أما في حالة التطبيقات متعددة المشاريع فيجب فتح .sln لفتح كافة المشاريع المرتبطة.

هنا ملاحظه لمبرمجي الفيجوال 6 أن الامتداد .sln يساوي الامتداد .vbg في الفيجوال 6 وكذلك مع إن فيجوال 2008 يستخدم امتداد أخر للملفات يختلف عن فيجوال 2002، 2003، 2005 لكن امتدادات ملفات 2002، 2003، 2005 لازالت تعمل تحت فيجول 2008.

أدوات الفيجوال بيسك 2008

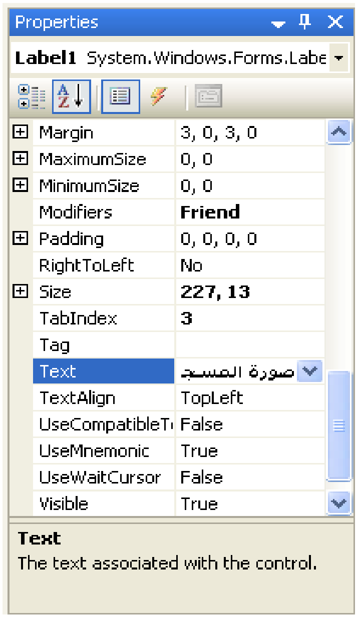
هناك العديد من الأدوات التي سوف تراها في فيجوال بيسك 2008 ومن ضمنها المصمم Designer ، متصفح المشروع Solution Explorer، نافذة الخصائص Properties Windows، صندوق الأدوات Toolbox، هناك الكثير من الأدوات التي تتشابه مع الأدوات الموجودة في الــ Word والــ Excel وهناك العديد من الأدوات سنتكلم عنها لاحقاً كل في وقته.

> مايكروسوفت قامت بإخفاء الأدوات النادرة الاستعمال ولكنك يمكنك إظهار الأداة التي تحتاجها من View ثم Toolbars واختيار الأداة التي تريد إظهارها.

نافذة الخصائص:

تستخدم نافذة الخصائص لتغيير الخصائص والإعدادات للكثير من الأزرار والمكونات الموجودة على الفورم. تستطيع تغيير هذه الإعدادات من نافذة الأدوات أو من خلال بعض الكود في خانة الكود (الإعدادات المغيرة من خلال نافذة الأدوات تعمل مع التطبيق بداية تشغيله ولكن الإعدادات التي من خلال الكود فيتم تغييرها بحسب الكود المضاف خلال فترة تشغيل البرنامج فيمكنك تغيير حجم الفورم أو طريقة إظهاره أو اختفاؤه أو لون الخط أو غيرة من الخصائص).

لنرى ألان نافذة الخصائص الخاصة بالأداة Label1 (طبعاً تظهر بعد الضغط على Label1)



لتغيير خاصية معينة في برنامجنا السابق AlMasjid Alharam قم بفتح المشروع ثم اظهر الفورم الرئيسي للمشروع بعدها اختار Label1 على الفورم ثم اذهب إلى الخصائص Properties انزل قليلا في نافذة الخصائص ستجد الخاصية Font (كمثال) اختر الزر المقابل لها سيفتح لك نافذة صغيرة مقاربة للتي بالــ Word تختار فيها نوع الخط ومقاسه بعد اختيار التعديلات التي تعجبك اختار OK ستلاحظ تغير نوع الخط وحجمه على حسب التغييرات التي عملتها في نافذة الخصائص في خاصية الخط. تستطيع كذلك التعديل على العديد من الخصائص التابعة للأداة Label ومن ضمن هذه الخصائص Cursor والتي تحدد كيف يتحول سهم الماوس في حالة مروره فوق الــ Label وكذلك الــ Location والتي تحدد مكان الــ Label على الفورم وغيرها من الخصائص. عندما نختار الليبل Label ستلاحظ صعوبة إمكانية تغيير مساحته (طوله وعرضه) على الفورم وعليه فيجب علينا أن نذهب إلى نافذة الخصائص ونضبط الخاصية AutoSize على False.

نعود للملاحظة التي تكلمنا عنها سابقاً وهي إمكانية التعديل على خصائص أي أداه خلال مرحلة تنفيذ البرنامج وذلك من خلال الكود:

انقر على زر Button2 والذي هو زر إظهار الصورة نقرتين Double-Click ليظهر لك مكان الفورم الخاص بالنقر على الزر Button2 ثم قم بإضافة هذا الكود:

```
' هذا الكود لتغيير خصائص الليبل
' هذه لتعديل لون الكلام الموجود بالليبل
Label1.ForeColor = Color.Blue
' هذا الكود لتغيير الكلام الموجود بالليبل
Label1.Text = "تم إظهار الصورة المطلوبة"
```

لاحظ أن السطر الأول والثاني والرابع لا علاقة لهم بالتنفيذ أو لا علاقة لهم بالأوامر لأنهم قد سبقوا بالبادئة ' وهي علامة التنصيص باللغة الانجليزية والموجود مع الحرف العربي ط على نفس الزر.

وعلية فان كل كود تسبقه ' فان الكون يرجع كلام لا علاقة له بالأوامر البرمجية وإنما فقط لتذكير المبرمجين عندما يريدون مراجعة الكود لاحقاً. نعود للكود أعلاه السطر الثالث عبارة عن أمر لتغيير الخط المكتوب بداخل الليبل إلى اللون الأزرق. أما السطر الأخير فهو لتغيير الكلام الموجود بداخل الليبل إلى "تم إظهار الصورة المطلوبة"

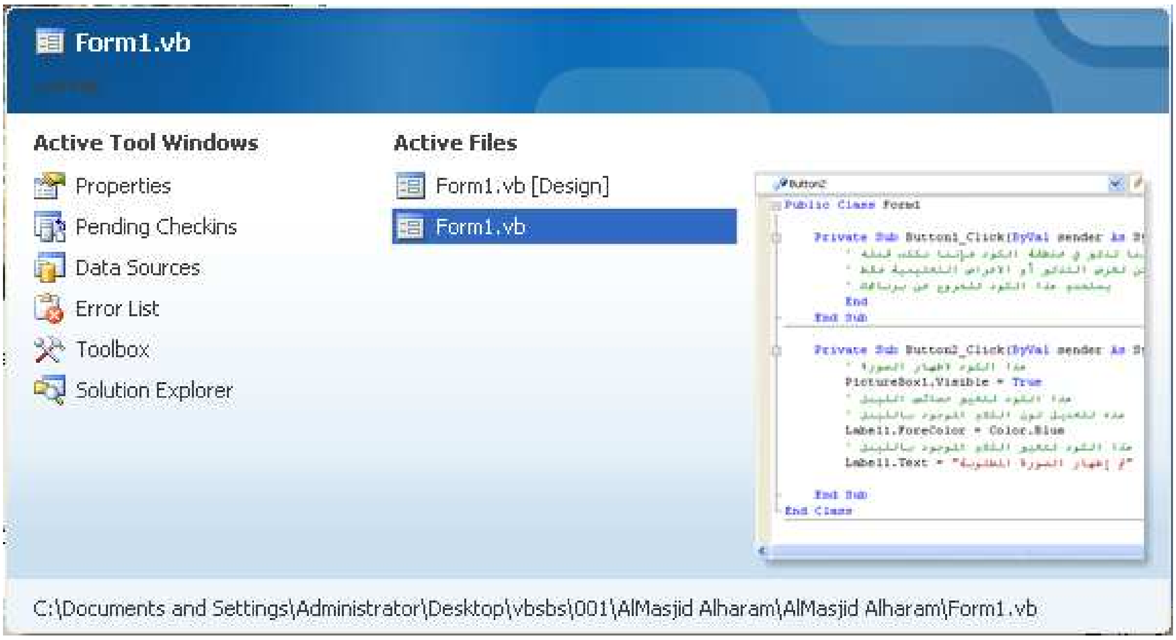
> يوجد طريقتين لعرض البنود في نافذة الخصائص الطريقة الأولى Alphabetical وتقوم بترتيب البنود A-Z والثانية Categorized وتقوم بترتيب البنود في نافذة الخصائص كمجموعات على حسب العلاقة فيما بينها. بحسب المؤلف إذا كنت مبتدءاً في فيجوال بيسك فالأفضل عرض البنود كمجموعات.

لتغيير مكان نافذة الخصائص إلى مكان أخر في بيئة التطوير أو سحب أي أداه من الأدوات البرمجية من مكان إلى أخر عليك القيام بالتالي : قم بعمل Right-Click على أعلى نافذة الخصائص ثم اختار احد الخيارات الخمسة :

Floating	مطفي: وتكون النافذة كنافذة مطفيه تأخذها من مكان إلى أخر، لكنها تشكل نوعا من الازدحام.
Dockable	قابل للرسو: بحيث يمكنه من الرسو (كالسفينة) في الأربع الجهات على الشاشة (يمين، شمال، فوق، تحت) وهو الخيار المفضل ويأتي افتراضي مع الفيجوال بيسك. اختر هذا الخيار ثم حاول تغيير مكان نافذة الخصائص ماذا تلاحظ: تلاحظ ظهور أربعة أسهم في الأربع الجهات في بيئة التطوير اختر أي واحد منهم تجد انتقال نافذة الخصائص إلى نفس الجهة.
Tapped Document	نموذج التاب: عندما تنتقل من الواجهة التصميمية للفورم إلى واجهة الكود فأنت تستخدم خيار التاب
Auto Hide	الإخفاء الأوتوماتيكي: يتم إخفاء الأدوات التي لا تستخدمها أوتوماتيكيا وعندما تقترب منها أو من مكانها فإنها تظهر.
Hide	مخفي: ويتم إخفاء أشرطة الأدوات التي لا تحتاجها ولا تظهر إلا بعد اختيار إظهار لها.
حاول تطبيق الثاني والرابع أكثر من مرة في أكثر من جهة من بيئة التطوير لكي تفهمهما بطريقة صحيحة.	

قد تقوم بالعديد من الإعدادات في أشرطة الأدوات والكثير من التعديلات التي قد تحتاجها إذا انتقلت إلى جهاز أخر، فيجوال 2008 يقوم بتصدير الإعدادات إلى ملف معين يمكنك أن تنقله بواسطة الفلاش أو أي من وسائط النقل إلى أي جهاز أخر وتقوم باستيراد الإعدادات. وللقيام بذلك اذهب إلى Tools < Import & Export Setting ثم اتبع بقية التعليمات لتصدير الإعدادات أو استيرادها إلى أو من ملف معين. إذا كنت قد تلاعبت في الإعدادات بطريقة غير صحيحة فيمكنك استعادة الإعدادات الأصلية باختيار Reset All Settings.

التنقل بين الملفات المفتوحة والأدوات البرمجية كنافذة الأدوات وبين الفورم الرئيسي للتطبيق الذي تقوم بتصميمه يتم بسهولة في فيجوال 2008، باستخدام Ctrl + Tab أنظر الصورة:



استمر بالضغط على Ctrl وارفع عن Tab وأرجعها من جديد تلاحظ سهولة التنقل بين الأدوات والفورمات في حالة وجود أكثر من فورم. وبينما أنت ضاغط على Ctrl اضغط الأسهم في الكيبورد للتنقل بين الأدوات المفتوحة والملفات المفتوحة، وكذلك تستطيع اختيار العنصر المراد اختياره بواسطة الماوس خلال الضغط على Ctrl.

تستطيع التنقل بين الأدوات في فيجوال 2008 بواسطة الضغط على Alt+F7.

إضافة Shift إلى Ctrl+Tab أو إلى Alt+F7 يجعل الحركة عكسية (من الخلف إلى الأمام. أي من العنصر الأخير إلى الأول.

فتح صفحة انترنت من خلال بيئة التطوير:

تستطيع فتح صفحة انترنت من خلال بيئة التطوير بدلاً من فتح المتصفح Explorer أو الفايرفوكس بدون الحاجة للخروج من بيئة التطوير من خلال الأتي:

View > Other Windows > Web Browser

أو بضغط الاختصار Ctrl+Alt+R

طبعا يقوم بإظهار متصفح بطريقة Tapped ويمكنك تغييرها إلى Floating أو Dockable أو حسب ما تشاء بواسطة Right-Click أعلى النافذة ثم اختر ما تشاء.

تستطيع تغيير الصفحة الافتراضية في المتصفح الذي يظهر في بيئة التطوير إلى الصفحة التي تريدها. من القوائم التابعة لبيئة التطوير : Options > Tools > ثم اختر Show All Options ثم أذهب إلى Environment من خلالها اختر Web Browser ثم قم بتغيير النص تحت Home Page إلى الصفحة التي تريدها.

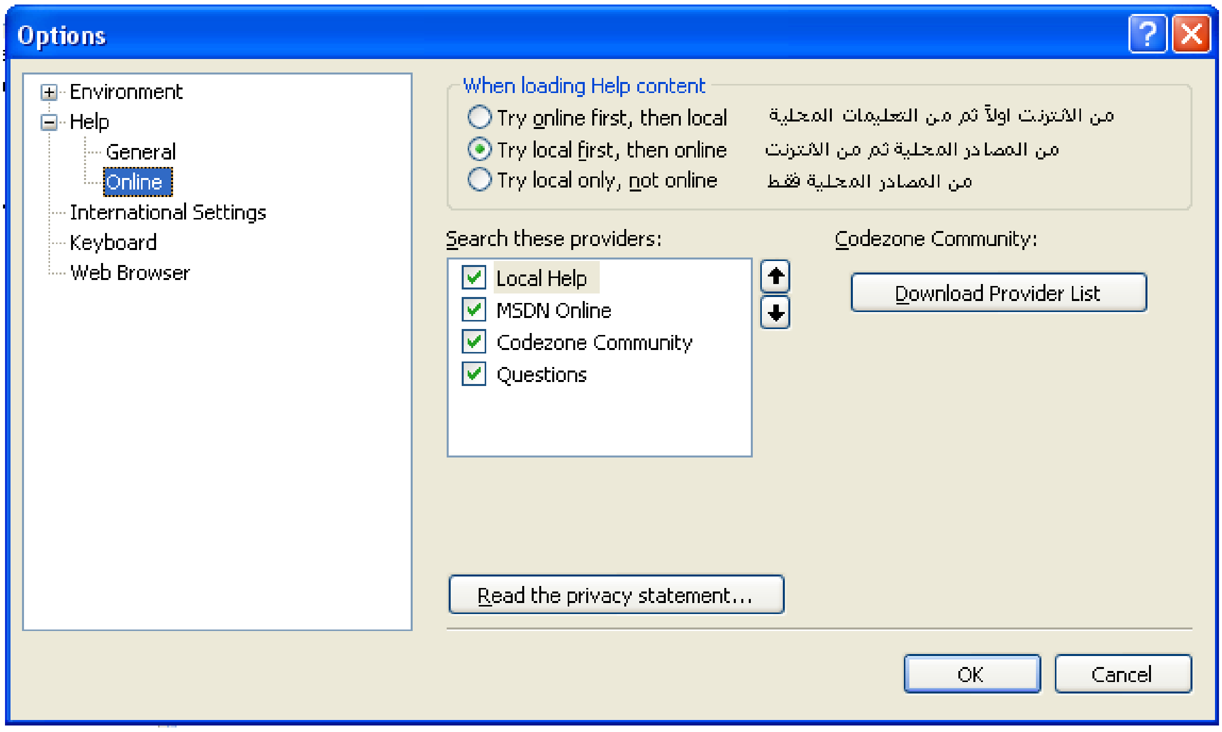
الحصول على التعليمات:

لا يخفى أهمية التعليمات التي قد نحتاجها خلال عملية تصميم تطبيق معين، فحتى المبرمجين المحترفين يحتاجون لبعض التعليمات لتنفيذ بعض المهام الصعبة. وعليه قد وفرت مايكروسوفت مكتبة التعليمات الـــ MSDN (طبعاً متوفرة باللغة الانجليزية) وهي مكتبة كبيرة جدا وهائلة وفيها

العديد من المعلومات التي تغني عن البحث هنا وهناك، وتوجد هذه المكتبة تحت قائمة Help المتوفرة ضمن بيئة التطوير ولكن قد تحتاج لتعليمات أخرى من مصادر خارجية عبر الانترنت عبر صفحات الـــ MSDN Online, MSDN news groups، والعديد من الصفحات والمواقع التعليمية المدعومة من شركة مايكروسوفت والتي تدعى CodeZone Community .

ما عليك إلا أن تضبط التعليمات الخاصة ببرنامج الفيجوال 2008 ليتقبل التعليمات من الانترنت (إذا كان لديك خط انترنت) ومن المصادر المحلية (التعليمات المتوفرة في الـــ MSDN المحلية) اذهب إلى Help ثم How Do I سيقوم بفتح نافذة التعليمات الخاصة بالفيجوال 2008 اذهب إلى

Tools ثم Options سيفتح نافذة اختر منها Help ثم Online



نختار نوع التعليمات التي نريد أن نحصل عليها، انظر اخترت الخيار الأوسط Try local first, then online وهو البحث في التعليمات المحلية أولاً (لان الانترنت عندي ضعيف) وإذا

لم يوجد يتم البحث في التعليمات التي في الانترنت طبعاً الأفضل هو جعل البحث في الانترنت أولا. قم باختيار طريقة الوصول للتعليمات التي تريدها.

إضافة التعليمات الهامة للمفضلة:

في حالة هناك تعليمات هامة وجدتها في مركز التعليمات وتريد أن تحتفظ بها حتى لا تبحث عنها لتجدها بصعوبة فيجوال 2008 يوفر لك خاصية الإضافة للمفضلة نفس التي موجودة في Internet Explorer فيمكنك إضافة أي تعليمات هامة للمفضلة والرجوع إليها لاحقاً:



بالإضافة إلى المقالات الهامة التي تستطيع أن تحتفظ بها في المفضلة تستطيع أن تحتفظ بنتائج البحث الهامة في المفضلة أيضا.

كيفية استعمال التعليمات:

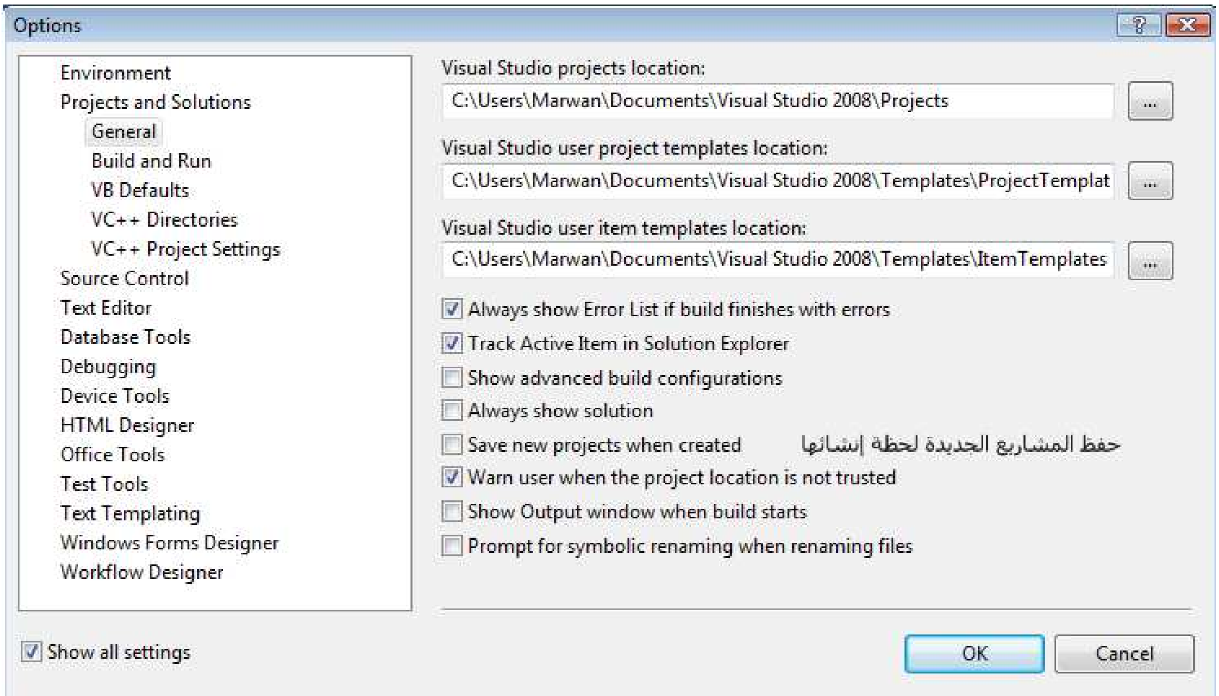
إذا أردت الحصول على معلومات	قم بعمل
مرتبة بواسطة مهمة البرمجة	اختر Help ثم How Do I
عن الميزة أو الأوامر التي تستخدمها حالياً	اختر Help ثم Dynamic Help
عن مواضيع	اختر Help ثم Content
بينما أنت تعمل تحرير للكود	اضغط الأمر الذي تريد معرفة معلومات عنه ثم اضغط F1

بينما أنت في صناديق الحوار	اضغط زر التعليمات (علامة الاستفهام) (مثل صناديق الحوار التي تظهر عندما تختار Tools ثم Options)
بالبحث عن كلمة معينة	تحت قائمة Help اختر Search ثم اكتب الكلمة التي تريد البحث عنها ثم قم بعملية الفلتره للنتائج باختيار لغة الفيجوال بيسك وغيرها من طرق الفلتره حسب الحاجة.
بالبحث في الانترنت	من قائمة Help اختر MSDN Forums
التواصل مع مايكروسوفت	من قائمة Help اختر Technical Support

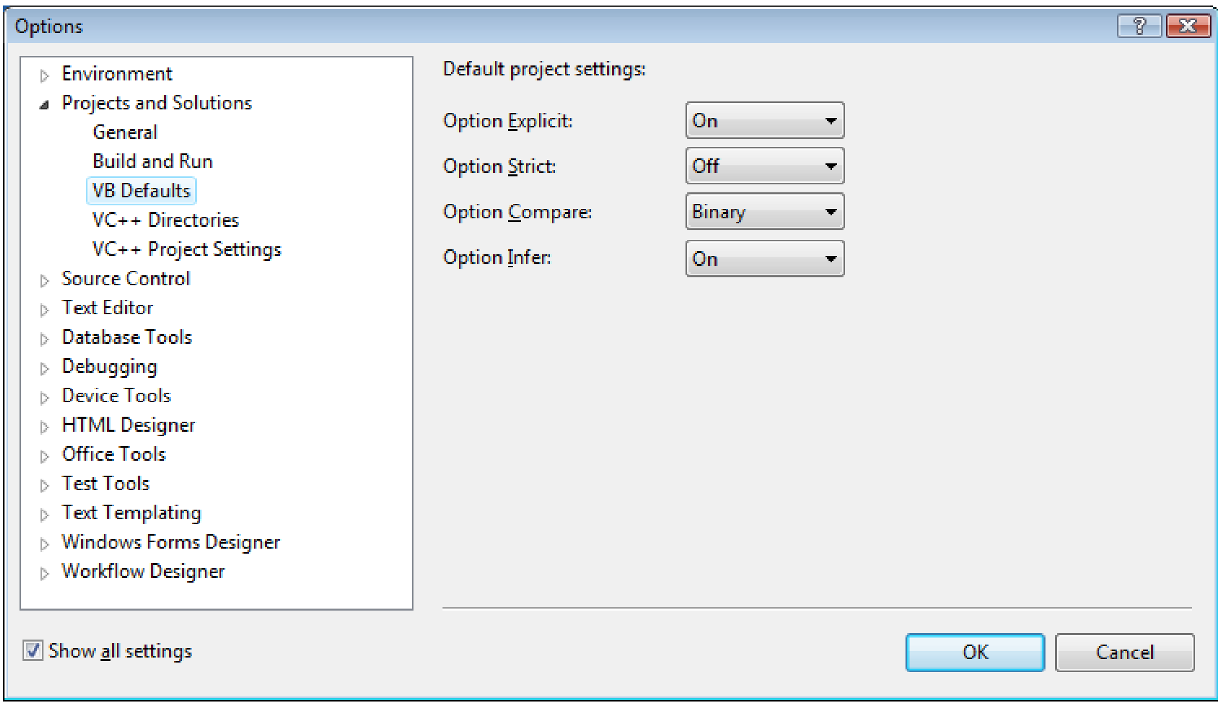
إعدادات هامة يجب التعرف عليها

قبل البدء في كتابة برنامجك الأول تبقى عليك معرفة بعض الإعدادات الهامة في الفيجوال بيسك 2008 عن طريقة حفظ التطبيقات التي تقوم ببرمجتها وكذلك عن خيارات التعريفات والأخطاء:

اذهب إلى Tools ثم اختر Options ومنها Projects and Solutions ثم اختر General : تأكد من انه ليس مؤشر على حفظ المشاريع الجديدة لحظة إنشائها لتكون النافذة كالتالي: (هذا سيوفر عليك بعض من الوقت بدلاً من حفظ المشروع من أول لحظة)



بعد الاختيار السابق هناك خيار هام، يتعلق بالــ Compiler أو المترجم إلى لغة الجهاز، اختر VB Defaults سيظهر لديك أربعة خيارات كالشاشة التالية:



سوف يتم شرح الأربع الخيارات بالتفصيل في الفصول اللاحقة، لا بد من ضبط الخيارات كالأتي:

Option Explicit : On, Option Strict: Off، تثبيت Option Explicit على الخيار On يلزمك بتعريف المتغيرات قبل استخدامها في التطبيق وهذه الميزة هامة جداً حتى لا ننسى من تعريف المتغير ونقوم ببناء التطبيق وفية أخطاء. Option Strict: Off يسمح للمتغيرات والكائنات عديدة الأنواع أن تتجمع بنوع واحد في حالات خاصة (كأن يتم تعريف رقم على انه متغير نصي بدون أخطاء) هذه ليست عادة حميدة في البرمجة ولكن في حالات خاصة قد تستلزم الضرورة القيام بذلك. سنعرف الكثير عن Option Compare في الفصل الثالث عشر. أما بالنسبة لـــ Option Infer فتعتبر خاصية جديدة في فيجوال بيسك 2008 فإذا ضبطت Option Strict على Off و ضبطت Option Infer على On وعرفت متغييرات بدون تحديد نوعية البيانات التي تمثلها هذه البيانات (فيجوال بيسك سوف يقوم بالتفكير لتحديد نوعية البيانات) سوف تعرف أكثر عن هذه الميزة في الفصل الخامس.

بشكل عام أنصحك بأن تضبط Option Infer على Off لتتجب النتائج الغير متوقعة خلال تنفيذ البرنامج بسبب المتغيرات.

الخلاصة: (بعد هذه الخلاصة سننتقل إلى الفصل الثاني)

من اجل أن	قم بالتالي
تفتح الفيجوال 2008	Start > All Programs > Microsoft Visual Studio 2008 > ثم اختر الأيقونة Microsoft Visual Studio 2008.
فتح مشروع موجود مسبقا	بعد فتح الفيجوال 2008 تحت قائمة File اختر Open Project أو قد تجد اسم المشروع على القائمة الرئيسية لآخر المشاريع المفتوحة فاختاره.
تنفيذ البرنامج تحويله إلى برنامج exe	من قائمة Debug اختر Start Debugging أو ضغط على F5

تغيير الخصائص	اختر المادة المراد تغيير خصائصها (حددها على الفورم) ثم اذهب إلى نافذة الخصائص واختر الخاصية التي تريد أن تغيرها ثم غيرها.
التنقل بين الملفات المفتوحة	بواسطة الضغط على Ctrl+Tab ثم انتقل بينها بتكرار الضغط على Tab أو تنقل بالأسهم بين الملفات المفتوحة وأدوات التطوير. وتستطيع اختار المادة المعنية بواسطة الماوس خلال الضغط على Ctrl+Tab
التنقل بين أدوات التطوير	بالضغط على Alt+F7 للتنقل بينها أو Alt+Shift+F7 للتنقل بطريقة عكسية.
تعديل طريقة الحصول على التعليمات	من قائمة Tools اختر Options
تعديل بيئة التطوير لتناسب العمل على فيجوال بيسك	من قائمة Tools اختر Import and Export Settings ثم اختر Reset All Settings ثم التالي ثم نعم Save my current settings ثم التالي. أخيرا اختر Visual Basic Developments Settings وثم زر Finish ثم اختر إغلاق.
تعديل خيارات بيئة التطوير	من قائمة Tools اختر Options ثم قم بتعديل خيارات بيئة التطوير على حسب المجموعات المذكورة (لتعديل خيارات المشاريع اذهب إلى General تحت خيار Projects And Solutions لتعديل خيارات المترجم إلى لغة الآلة الــ Compilerاذهب إلى VB Defaults تحت نفس المجموعة)
إغلاق الفيجوال 2008	تحت قائمة File اختر Exit

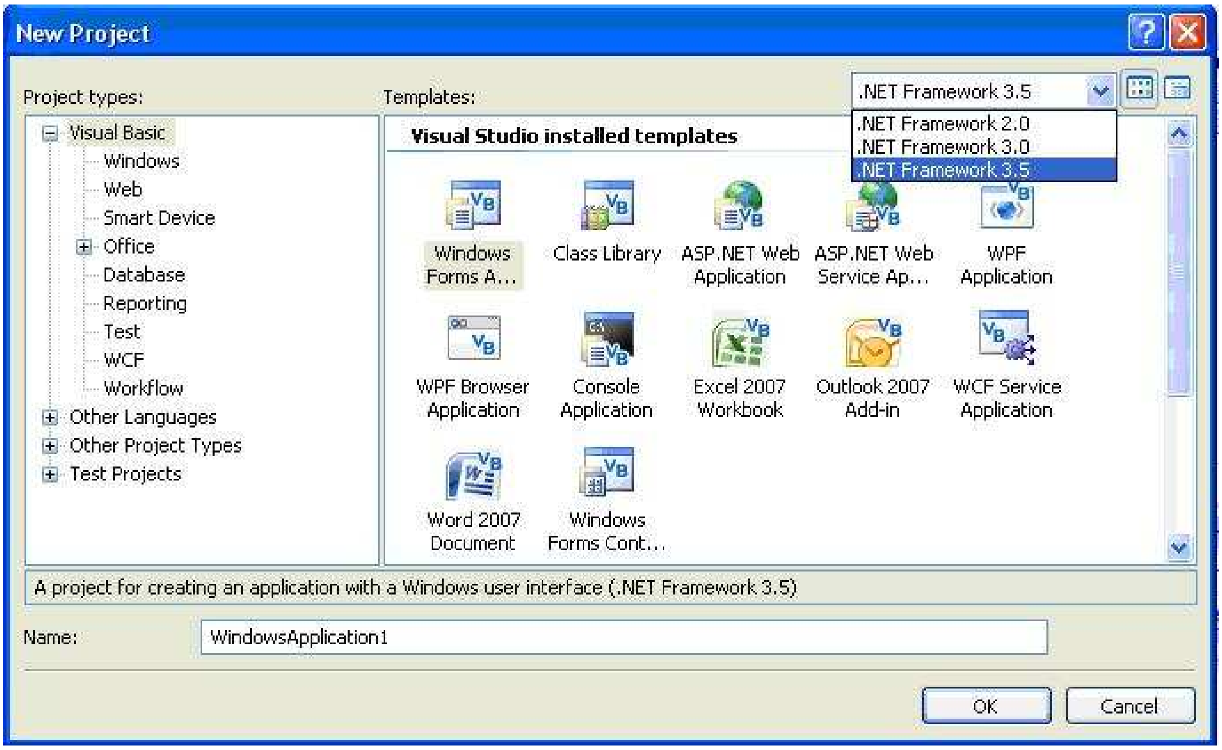
الفصل الثاني: كتابة برنامجك الأول:

تعرفت في الفصل الأول على بيئة التطوير والتعامل مع الأدوات البرمجية والقوائم التابعة للفيجوال وتغيير الإعدادات. في هذا الفصل ستقوم بكتابة برنامجك الأول. علينا أن نعرف أولا إن هناك نوعين من البرامج النوع الأول Windows-Based Programs البرامج التي تعمل تحت منصة الويندوز النوع الثاني Web-Based Programs البرامج التي تعمل على سيرفرات الانترنت. سنتعرف هنا على البرامج التي تعمل تحت منصة نظام الويندوز وستكتب برنامجك الأول ليعمل تحت بيئة ويندوز. البرنامج الذي سنقوم بكتابته هو برنامج يعتمد على توليد الأرقام العشوائية (توليد الأرقام العشوائية قد تحتاجه للعديد من المهام البرمجية) حتى إذا قام بتوليد رقم معين وليكن الرقم 5 أو 7 يقوم البرنامج بإظهار جملة معينة أو صورة. قم بفتح الفيجوال 2008. من قائمة File اختر New Project أو اضغط Ctrl+N سيفتح لك نافذة لاختيار لغة البرمجة التي تريد أن تصنع بها برنامجك اختر الفيجوال بيسك Visual Basic ثم اختر من القائمة اليمنى Windows Forms Application عند خانة Name في الأسفل اكتب اسم لبرنامجك وليكن هذا الاسم Arqam:

تستطيع إنشاء مشروع جديد بالنقر على كلمة Project مقابل Create



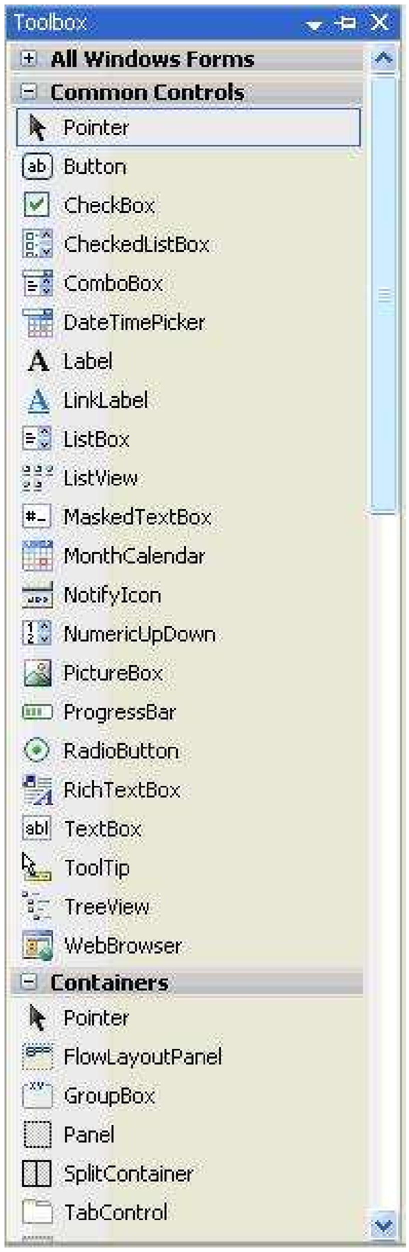
في حالة فتح مشروع جديد ستظهر لك نافذة الخيارات التالية:



تستطيع برمجة المشروع بحيث يعمل تحت .Net Framework 2.0 أو 3.0 أو 3.5 لكن إذا اخترت لمشروعك العمل تحت الفريم ورك نسخة رقم 2.0 لن تستفيد من الخيارات التي تقدمها النسخة الحديثة 3.5 فسيقوم فيجوال بيسك 2008 بإخفاء الخيارات الحديثة المتوفرة فقط في الفريم ورك الحديثة. وعلية فالأفضل أن تستخدم الفريم ورك رقم 3.5 ولكن في حالات معينة قد تضطر لاستخدام نسخ الفريم ورك القديمة فليكن هذا في حسبانك (ستجد العديد من التفصيل حول الدوت نت فريم ورك في الفصل الخامس من هذا الكتاب). ستلاحظ من النافذة أعلاه إمكانية اختيار لغات أخرى غير الفيجوال بيسك مثل الـــ C# وغيرها من اللغات المتوفرة تحت بيئة الفيجوال نت 2008.

بعد إضافة البرنامج اذهب ليسار الشاشة عند قائمة على اليسار يوجد صندوق الأدوات Toolbox هذا الصندوق يحتوي على كل الأدوات التي تحتاجها أو المضافات التي تحتاجها لبرنامجك فيمكنك إضافة أزار وصناديق نص وغيرها من المكونات التي تجعل مستخدم برنامجك يتفاعل معها: فمثلا

يقوم بالضغط على زر معين ليظهر له تاريخ اليوم أو نتيجة عملية حسابية أو غيرها. انظر هذا صندوق الأدوات:



نختار من صندوق الأدوات التالي:

عدد اثنين أزار Buttons

عدد أربعة ليبلات Labels

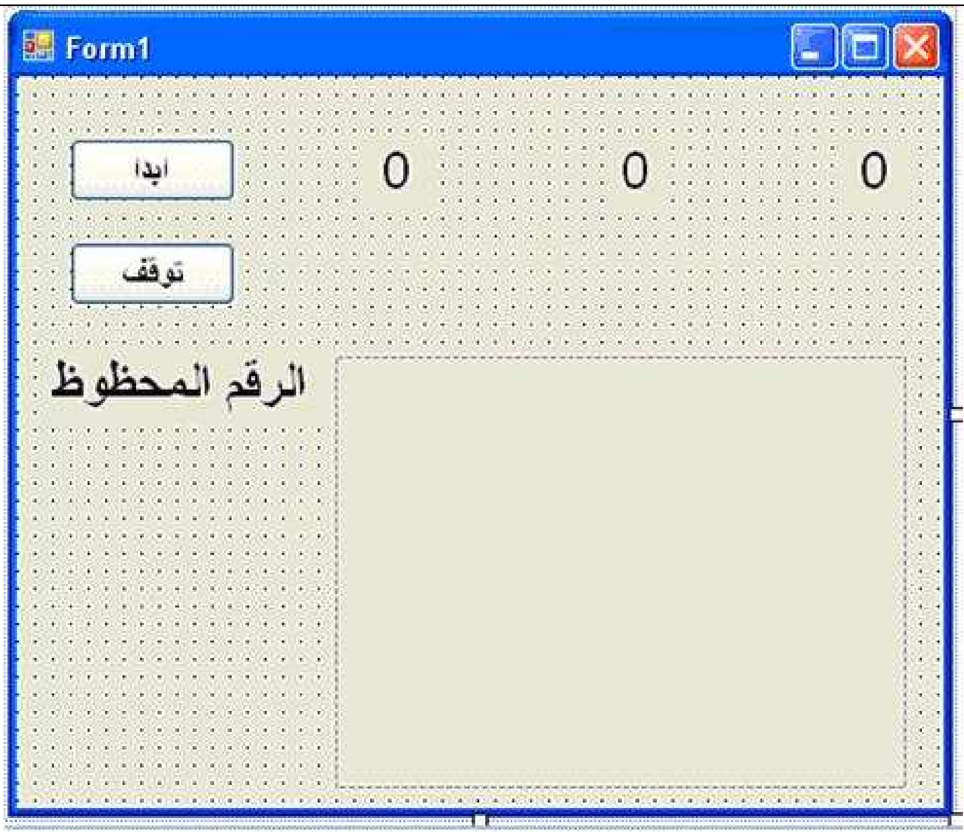
عدد واحد صندوق للصور Picture Box : قم بإضافة كل مكون من المكونات أعلاه كل على حدة وإذا رأيت أن الفورم أصبح ضيقاً فاذهب إلى أسفل يمين الفورم (عند الزاوية) في حينه سيتحول الماوس إلى سهم محدد من الطرفين، اضغط على الفورم وقم بعملية التكبير أو التصغير للفورم. لإضافة المكونات المذكرة أعلاه للفورم نذهب إلى Toolbox (صندوق الأدوات في الصورة أعلاه) ثم نذهب بالماوس إلى فوق الـــ Button نضغط عليه ثم نذهب إلى الفورم ونضغط عليه فنلاحظ انتقال الزر إلى الفورم لنقل الزر الثاني نستطيع تكرار العملية أو نستخدم طريقة النسخ واللصق فوق الفورم فنحدد الزر بواسطة الماوس ثم نضغط Ctrl+C ثم Ctrl+V

أما الطريقة الثالثة لإضافة المكونات إلى الفورم فهي الذهاب إلى صندوق الأدوات Toolbox وقبل اختيار المكون إلى الفورم نضغط على الزر Ctrl ثم نختار المكون المراد ثم نذهب إلى الفورم ثم نضغط في أي مكان في الفورم ونحن مواصلين على ضغط الزر Ctrl ثم نضغط مرة ثانية على الفورم ثم مرة ثالثة، سنلاحظ أن المكون الذي اخترته تم لصقه أكثر من مرة على الفورم، نستخدم هذه الطريقة في حالة إذا أردنا إضافة العديد من الأزرار أو المكونات إلى الفورم دفعة واحدة بدون الحاجة للذهاب إلى صندوق الأدوات عند كل مرة.

عند إضافة الأزرار إلى الفورم نستطيع أن نقوم بتغيير مكانهما في داخل الفورم وكذلك نستطيع أن نغير من حجمهما بداخل الفورم لتغيير المكان التابع لهما في داخل الفورم نحدد الزر أو المكون المراد تغيير مكانه بداخل الفورم ثم نقوم بعملية السحب والتحريك له بداخل الفورم وعلية فسنستطيع أن نغير مكانة إلى أسفل أو أعلى الفورم أو إلى الوسط. (في حالة إذا أردنا أن نقوم بسحب الاثنين الأزرار مرة واحدة إلى أي مكان في الفورم نقوم بتحديدهما الاثنين (أو كل الأزرار في حالة إذا ما كان هناك أكثر من اثنين)وذلك بالضغط على Ctrl واختيار الأول ومواصلة الضغط على Ctrl ثم اختيار الثاني وبعد ذلك نستطيع سحبهما إلى مكان معين في الفورم مع بعض ونستطيع كذلك من تغيير خصائصهما المشتركة مع بعض)

	نفس عمليات السحب والإلقاء التي تطبقها على الأزرار Buttons تستطيع أن تطبقها على بقية المكونات مثل الليبلات Labels وغيرها من المكونات.
	عند القيام بتغيير مكان الزر على الفورم وفي حالة ذهبنا إلى نهاية الفورم سنلاحظ تكون خط ازرق ليحدد لك انك اقتربت من نهاية الفورم وعند توفر العديد من الأزرار أو المكونات على الفورم ستلاحظ تكون الخط الأزرق ليبين لك إن الزر أو المكون الذي تقوم بتحريكه يوازي بقية المكونات على الفورم

	تستطيع إظهار الشبك (Grid) وهي خطوط عمودية وأفقية على الفورم تستطيع أن تستخدمها لمعرفة هل الأزرار أو المكونات على خط متوازي أو لا بمعنى آخر أنها تساعد في تنظيم المكونات على الفورم. في الفيجوال 2003 وكذلك الفيجوال 6 كانت الـــ Grid أو الشبك تظهر بشكل افتراضي على الفورم إذا كنت تريد إظهارها اذهب إلى قائمة Tools بعدها Options ستظهر لك نافذة تأكد من اختيار Show All Settings أسفل النافذة ثم اذهب إلى Windows Forms Designer اختر منها General ثم اضبط Show Grid على True واضبط LayoutMode على SnapToGrid أغلق الفورم ثم افتحه مرة ثانية سيظهر بهذا الشكل:



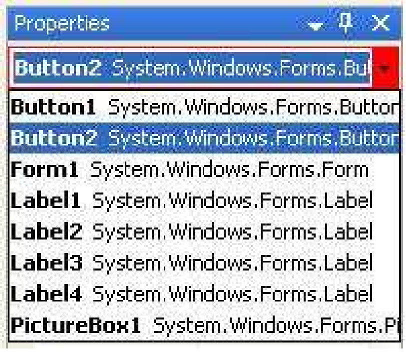
تستطيع تغيير حجم أي زر أي مكون على سطح الفورم بأن تقوم بتحديده سيظهر لك ثمانية مربعات بيضاء على الزر أو المكون المحدد تستطيع أن تضغط على احدها وتقوم بعملية تكبير أو تصغير للزر أو للمكون انظر الزر التالي: 

لاحظ الثمانية المربعات البيضاء.

تستطيع إلغاء أي زر أو أي مكون من سطح الفورم بتحديد المكون المراد ثم بالضغط على Delete، الآن قم بعملية إضافة المكونات إلى الفورم وإلغائها وتغيير مكانها على الفورم وكذلك تغيير حجمها لتعرف أكثر.

قمنا الآن بإضافة اثنين أزرار وأربعة ليبلات وواحد صندوق للصورة. بعد إضافة المكونات إلى الفورم قم ببعض التعديل على خصائص هذه المكونات لتتلائم مع حجم الفورم ونوع الخط المراد عرضة على شاشة المستخدم. (لإظهار نافذة الخصائص اضغط على F4)

لتعديل خصائص أي مكون علينا أن نحدد المكون المراد تعديل خصائصه ثم ذهب إلى نافذة الخصائص ونقوم بالتعديلات اللازمة أو من نافذة الخصائص نختار السهم أعلى نافذة الخصائص ونقوم بالتعديل علية لاحظ الصورة (قمت بتلوين مكان السهم بالأحمر):



يمكنك تعديل خصائص أكثر من مكون مرة واحدة وذلك بتحديد هذه المكونات مرة واحدة ثم تعديل خصائصها أو اختيار المكون الأول ثم الضغط على Shift ثم اختيار المكون الثاني وهكذا. للعلم في حالة اختيار العديد من المكونات ليتم تعديل خصائصها مرة واحدة فقط الخصائص المشتركة بين هذه المكونات ستظهر في نافذة الخصائص. ما نريده منك الآن هو أن تقوم بتغيير الخصائص للمشروع الذي نكتبه في هذا الفصل حاول تغيير الخط نوعه وحجمه طريقة عرضه وغيرها من الخصائص، سنقوم الآن بذكر بعض الخصائص الخاصة بالمكون Button أو الزر.

الخاصية	شرحها
(Name)	اسم الزر المختار وافتراضيا تسمى Button1,2,3 ولكن تستطيع تغييرها للتناسب مع طبيعة الزر المراد تكوينه فممكن تسميتها إبدأ StartBtn أو إنهاء ExitBtn أو غيرها من التسميات والتسمية مهمة في مرحلة الكود لنعرف لماذا هذا الزر.
Anchor	تحدد مكان الزر في حالة تكبير الفورم وقت التشغيل (المقصود بوقت التشغيل وقت تشغيل البرنامج) فبإمكانك اختيار احد الأسهم أو اثنين أو

	الثلاثة أو الكل. افتراضيا اليسار والأعلى يكون مختار مسبقا وعلية ففي حالة تكبير الفورم وقت التشغيل فان الزر يتجه نحو أعلى يسار الفورم. أما في حالة اختيار الأربع الجهات ففي حالة تكبير الفورم فان الزر يتجه نحو وسط الفورم.
AutoEllipsis	في حالة ضبطها على True وإذا كان النص داخل الزر اكبر من حجم الزر فستقوم هذه الخاصية بإضافة ثلاث نقاط في الرز أما إذا كان بنفس مساحة الزر فلا تضيف شيئاً.
AutoSize	تسمح هذه الخاصية في حالة ضبطها على True بتكبير حجم الزر إذا كان النص اكبر من حجم الزر فيكبر الزر ليسمح برؤية النص كاملاً.
AutoSizeMode	طريقة التعامل مع النص الكبير GrowOnly يكبر الزر بمقدار النص GrowAndShrink يكبر الزر في نفس الوقت الذي يتم فيه تصغير النص.
BackColor	لون الزر.
Background_ Image	خلفية الزر حيث تستطيع أن تضع صورة كخلفية في الزر.
Background_ ImageLayout	طريقة عرض الصورة في حالة اختيار صورة كخلفية.
Cursor	شكل السهم وقت التشغيل وقت مروره على الفورم ستلاحظ العينات.
Dock	طريقة عرض الزر على الفورم.

Enabled	في حالة True يكون الزر طبيعياً في حالة False يكون الزر مرئي ولكن لا يمكن التعامل معه.
FlatStyle	أربع خيارات لعرض الزر وهذه تعتمد على ذوق المصمم وكذلك على نوعية البرنامج جربهم واحده بعد الأخرى ثم قم بتشغيل التطبيق ولاحظ تغيير شكل وطراز الزر.
Font	سيظهر لك نافذة كالتي في الورد لتغيير حجم ونوع وشكل الخط.
ForeColor	لتغيير لون الخط.
Location	مكان الزر على الفورم بتغيير الأرقام يتغير المكان أفقيا وعمودياً.
Locked	نستفيد من هذه الخاصية وقت تصميم التطبيق وتقوم بتثبيت مكان الزر على الفورم حتى لا يتغير مكانه إذا عبثنا بالماوس أو أخطأنا.
MaximumSize	اكبر مساحة ممكنة للزر، تستطيع تحديد اكبر عرض width ممكن للزر وكذلك اكبر ارتفاع Height ممكن للزر.
MinimumSize	عكس الأول.
RightToLift	اتجاه الزر (من اليمين إلى اليسار) مفيد للتطبيقات ذوو الواجهة العربية.
Size	مساحة الزر العرض Width والارتفاع Height
TabIndex	بعد تشغيل التطبيق تستطيع التنقل بين المكونات بواسطة زر الكيبورد Tab أما الخاصية TabIndex فتحدد رقم الزر في حالة الضغط على Tab فالرقم صفر يبين انه بعد تشغيل البرنامج مباشرة سيكون Tab فوق هذا الزر وإذا تم ضغط زر الكيبورد Tab فسينتقل إلى الزر ذو

	رقم TabIndex 1 وهكذا.
TapStop	في حالة الضبط على False فعند تشغيل التطبيق والضغط على زر الكيبورد Tab فلن ينتقل إلى هذا الزر لأنك ضبطت القيمة على False وإذا أردت أن ينتقل إلى هذا الزر فاضبط القيمة على True
Text	النص الموجود داخل الزر. جرب اكتب اسمك أو ما تريد في هذه الخاصية ولاحظ ماذا سيتغير على الزر.
TextAlign	طريقة عرض النص بداخل الزر.
Visible	في حالة ضبط هذه الخاصية على False فإن الزر سيختفي في حالة تشغيل البرنامج.
هناك العديد من الخواص التي لم اذكرها للعديد من الأسباب ولكن الخصائص المذكورة أعلاه تعتبر الأكثر استخداما وهي نفسها خصائص مشتركة بين العديد من المكونات.	

عندما استخدم مصطلح مكون فاقصد به زر Button أو ليبل Label أو صندوق نص TextBox أو صندوق صورة PictureBox أو غيرها من المكونات التي تظهر في صندوق الأدوات.

بالنسبة لصندوق الصورة فبعد تحديده سترى سهم صغير أعلى يمين الصندوق قم بالضغط على السهم ثم اختر Chose Image ستفتح لك نافذة اختر Local resource ثم اختر زر Import ثم اختر صورة ستلاحظ انتقال الصورة إلى صندوق الصورة على الفورم تركت لك صورة في الملف رقم 002 باسم sky قم باختيار الصورة، طبعا بعد اختيار الصورة اذهب إلى خصائص صندوق الصورة واضبط الخاصية Visible على False. تستطيع الضغط -Double

Click على الخاصية فتتغير بين True و False لأنها قيمة Boolean وكل قيمة بوليان موجودة في الخصائص تستطيع تغييرها بالضغط دبل كليك على الخاصية المراده. لنعرف الآن ماذا نريد من البرنامج الذي نقوم بتصميمه الآن شاهد الصورة التالية لتعرف ماذا نريد من



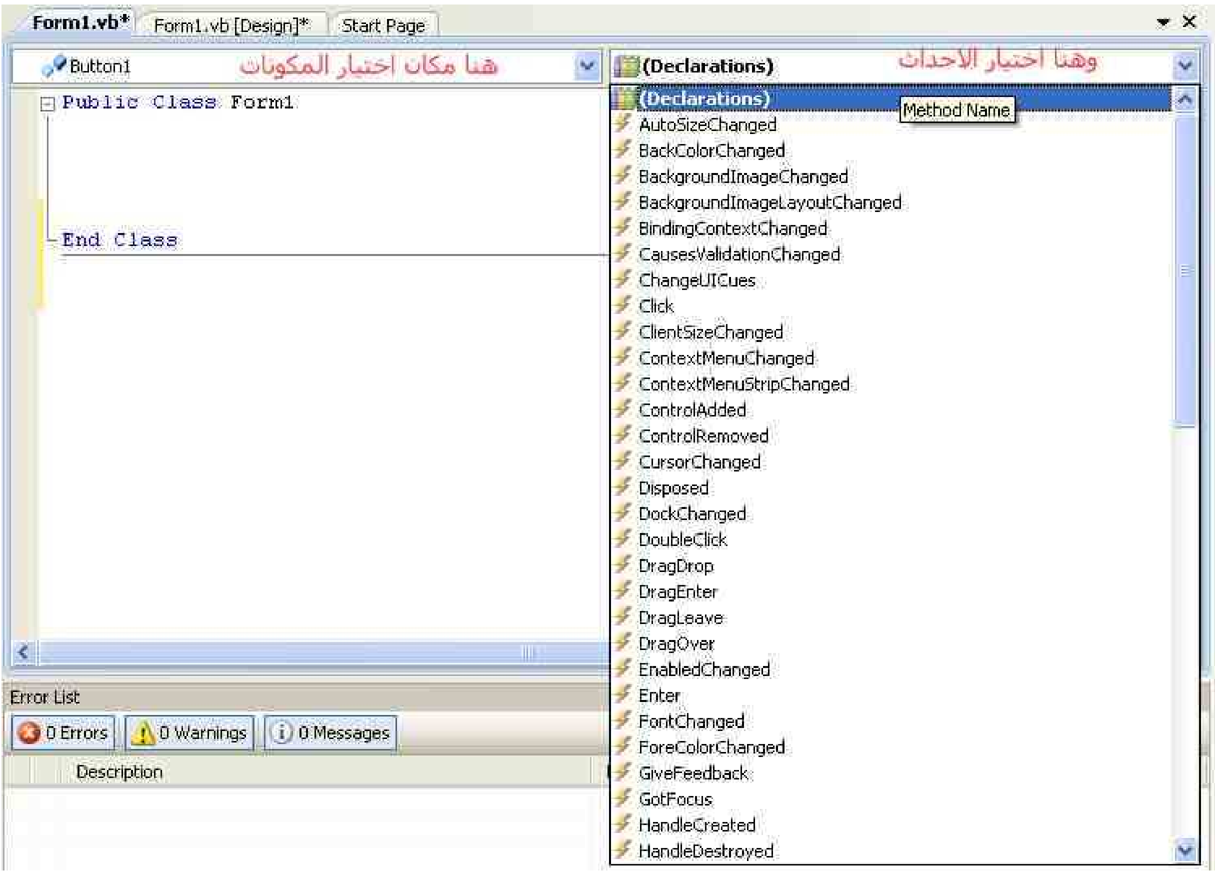
البرنامج:

لدينا اثنين من الأزرار زر ابدأ وزر توقف وصندوق للصورة فإذا ضغطنا الزر ابدأ يقوم البرنامج بتوليد الأرقام العشوائية في الثلاثة الليبلات وإذا ضغط الزر توقف فان البرنامج يتوقف عن توليد الأرقام العشوائية فإذا كان بين هذه الثلاثة الأرقام التي تم توليدها الرقم 7 ستظهر الصورة في صندوق الصورة.

مرحلة الكود:

في هذه المرحلة سوف نعرف كيف يتم توليد الأرقام العشوائية وكيفية إظهار الصورة في حالة وجود الرقم 7 في احد الليبلات. لننتبه أن الزر الذي سيبدأ في توليد الأرقام العشوائية هو الزر ابدأ والحدث الذي نريد أن يتسبب في توليد الأرقام العشوائية هو حدث النقر على الزر أو الحدث Click كيف نعرض هذا الحدث. نستطيع ذلك بأكثر من طريقة الأولى النقر المزدوج -Double

Click على الزر في الفورم سينتقل لك مباشرة إلى حدث النقر على الزر. الطريقة الثانية هي عمل Right-Click على الفورم ثم اختر View Code ثم أعلى منطقة الكود اضغط السهم المنزلق ثم اختر الزر الذي تريد تعديل خصائصه ثم اذهب إلى اليمين تحت Declarations اضغط السهم ثم اختر الحدث Click انظر الشكل التالي لمعرفة كيفية الوصول إلى الحدث المراد:



بعد الوصول إلى مكان الحدث التابع للزر Button1 والذي هو زر ابدأ (نستطيع تغيير الاسم للزر ليتلائم مع وظيفة الزر بداخل التطبيق كما هو موضح في شرح جدول الخصائص) نقوم بكتابة الكود التالي بين الجملتين Private Sub و End Sub:

```
PictureBox1.Visible = False
    Label1.Text = CStr(Int(Rnd() * 10))
    Label2.Text = CStr(Int(Rnd() * 10))
    Label3.Text = CStr(Int(Rnd() * 10))
```

```
        If (Label1.Text = "7") Or (Label2.Text = "7") Or (Label3.Text = "7 ")
_
Then PictureBox1.Visible = True
```

ملاحظات حول الكود السابق:

للانتقال من السطر الأول إلى السطر الذي يليه قم بالضغط على Enter. أما في حالة السطر الطويل فيمكنك تقصيره بإضافة _ ثم الضغط Enter بعدها مباشرة لتنتقل إلى السطر الثاني.

خلال عملية كتابة الكود ستلاحظ أن بيئة التطوير تساعدك على كتابة الكود بواسطة الإكمال التلقائي، وكذلك ستلاحظ تلوين بعض الكلمات فاللون الأزرق للكلمات المحجوزة في بيئة التطوير لأوامر البرمجة والتطوير وعلية فلا يمكن تسمية المكونات بهذه الكلمات ولا يمكن استخدامها كمتغيرات في البرنامج، أما اللون الأخضر فهو للجمل التذكيرية التي تستخدمها لتتذكر لماذا كتبت هذه الجملة البرمجة وهذه الجمل لا تدخل في عملية الكود ولا يتم ترجمتها إلى لغة الآلة ولكن تستخدم للتذكير فقط. أما اللون الأحمر فسيستخدم للنصوص Texts فإذا كان لدينا نص معين في الكود فستقوم بيئة التطوير بتلوينه باللون الأحمر.

قم الآن بإضافة الكود للزر توقف:

قم بعمل Double-Click على الزر توقف الموجود على الفورم ستنقلك بيئة التطوير إلى الكود مباشرة وبالتحديد إلى الحدث Click وهو المطلوب، اكتب هذا الكود:

```
End
```

الكود أعلاه معناه أغلق التطبيق ففي حالة تشغيل التطبيق والضغط على الزر المعين سيغلق التطبيق. صورة لمنطقة الكود بعد إضافة الكود المطلوب:

```
Button2                                   Click

    Private Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles Button1.Click
        PictureBox1.Visible = False ' لإخفاء الصورة
        Label1.Text = CStr(Int(Rnd() * 10)) ' لإختيار الرقم العشوائي من صفر الى عشرة
        Label2.Text = CStr(Int(Rnd() * 10))
        Label3.Text = CStr(Int(Rnd() * 10))
        ' اذا كان الرقم العشوائي يساوي سبعة فقم بالتالي
        If (Label1.Text = "7") Or (Label2.Text = "7") Or (Label3.Text = "7 ") _
        Then PictureBox1.Visible = True

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e _
                              As System.EventArgs) Handles Button2.Click
        End
    End Sub
End Class
```

دعونا الآن لنفكر مرة ثانية كيف يعمل هذا البرنامج بعد تشغيله: فبعد تشغيل البرنامج تكون الصورة الموجودة في صندوق الصورة مختفية وذلك لأننا ضبطنا الخاصية Visible لصندوق الصورة أوPictureBoxعلى False وحين الضغط على زر ابدأ أكثر من مرة يقوم التطبيق بتوليد أرقام عشوائية من صفر إلى عشرة

```
((Label1.Text = CStr(Int(Rnd() * 10
```

Cstr تساعد في التحويل إلى String أما Int فتعني أن الأرقام المولدة ستكون أرقاما صحيحة أما Rnd فتعني العشوائية Random حيث سيقوم البرنامج بتوليد الأرقام العشوائية، أما الرقم عشرة فهو حدود الأرقام العشوائية حيث يتم توليد الأرقام من 0 إلى 10 (جرب اكتب 100 بدل عشرة وشغل البرنامج وانظر كيف تتم عملية توليد الأرقام، حيث سيتم توليد الأرقام من 0 إلى 100) وإذا كان الرقم المولد في احد الليبلات الثلاثة يساوي 7 فسيقوم البرنامج بإظهار الصورة. أما في حالة الضغط على الزر توقف فسيتم إغلاق التطبيق. في الشكل أدناه سنوضح كيف يقوم الفيجوال بيسك بتنفيذ العملية:

الكود ((Label1.Text = CStr(Int(Rnd() * 10

الكود	التنفيذ
Rnd	0.7055475
Rnd * 10	7.055475
(Int(Rnd * 10	7
((CStr(Int(Rnd() * 10	"7"
((Label1.Text = CStr(Int(Rnd() * 10	يتم نقل الرقم 7 إلى الليبل1

وبعد ظهور الرقم سبعة في احد الليبلات يقوم البرنامج بإظهار الصورة في صندوق الصورة عن طريق الجملة الشرطية If، والتي تشترط وجود الرقم 7 في الليبل الأول أو الثاني أو الثالث.

لتشغيل البرنامج الذي تم تصميمه اضغط على F5، أو بالذهاب إلى قائمة Debug اختر Start Debugging.

عند تشغيل البرنامج لأكثر من مرة ستلاحظ أن الأرقام التي يتم توليدها متشابهه، جرب ذلك بتشغيل البرنامج أكثر من مرة وقم بتدوين الأرقام في الخانات الثلاث ستلاحظ تكرار الأرقام بين كل تشغيل والآخر للبرنامج. إذا ما الحل علينا أن نعرف إن الدالة Ran تقوم بتوليد نفس الأرقام العشوائية ولجعل هذه الأرقام تختلف بشكل عشوائي حقيقي، نضيف الكلمة Randomize للفورم عند الحدث Load وللوصول إلى الحدث نضغط Double-Click على الفورم فسينقلنا مباشرة إلى الحدث Load. أو الذهاب للحدث بقراءة أحداث الفورم والوصول إلى الحدث Load كما تعلمنا سابقاً في أحداث الزر Button1.

توجد نسخة من التطبيق الذي قمت بتصميمه في المجلد المجلدات المرفقة مع الكتاب رقم المجلد 002 يمكنك فتح التطبيق الذي تم تصميمه مسبقاً وعمل مقارنة بين التطبيق الذي قمت بتصميمه والموجود بالمرفقات.

نشر البرنامج:

إذا قمت بعملية برمجة لبرنامج معين وأردت أن تنشره فبعد حفظه اذهب إلى المجلد الموجود في نفس المسار الذي حفظت فيه المشروع وستجد البرنامج المصمم. انسخه ثم قم بنشرة على الأصدقاء لتجربته أو العملاء في حالة بيعه لكن يلزمك نشر الفريم ورك التي تم برمجه البرنامج بها فإذا استخدمت Framework 2 أو Framework3.5 فيلزم نشر نفس الفريم ورك مع البرنامج، كما يوجد بعد الإعدادات الخاصة برقم النسخة ورقم المراجعة تستطيع تعديلها من قائمة Project في نهاية القائمة ستجد (اسم مشروعك) Properties اخترها من النافذة التي تظهر اختر منها Application على اليسار ثم اختر منها Assembly Information وقم بتغيير رقم النسخة عند Assembly Version و File Version .

تنصيب برنامجك:

يتيح لك الفيجوال بيسك 2008 العديد من الخيارات لتنصيب برنامجك على الأجهزة الأخرى. بالمقارنة ففيجوال بيسك 6 يتطلب منك برنامج تنصيب معقد ليقوم بنقل مكتبات الربط الديناميكية والملفات المساندة للبرنامج وكذلك لابد من تسجيل البرنامج في نظام التشغيل ليتم تشغيله على الأجهزة الأخرى. بينما في فيجوال 2008 فيتم إعداد ملف التنصيب مع أول عملية تجربة للبرنامج أو مع أول عملية بناء للبرنامج طبعا البرنامج يحتاج للملفات المساندة وهي لغة مايكروسوفت الوسيطة MSIL و metadata (البيانات التي تصف المحتوى) و manifest (السجلات) و بقية الملفات والمكتبات المساعدة. إذا أردت نقل برنامجك من جهاز كمبيوتر إلى أخر عليك فقط بإتباع طريقة القص واللصق من جهاز إلى أخر بشرط وجود نسخة الفريم ورك المناسبة على الجهاز

الأخر وكذلك يمكنك استخدام طريقة Xcopy المتوفرة في الـــ command في الـــ DOS. لكن في الحقيقة إذا كان البرنامج مصمم لغرض تجاري أو لغرض عملي بحيث إن البرنامج مهم لدرجة ما فلا يمكن لنا أن نقوم بعملية النسخ واللصق أو عملية Xcopy وإنما من الأفضل عمل setup خاص بالبرنامج حتى إذا أردنا أن نحذفه من الجهاز نذهب إلى لوحة التحكم (إضافة وإزالة البرامج)، الفيجوال بيسك 2008 يوفر طريقتين لنشر وتوزيع برنامجك 1- ClickOnce 2- WindowsInstaller أما الطريقة الأولى فيمكنك استخدامها إذا أردت التسهيل مع المستخدمين لبرنامجك، حيث يمكن للمستخدمين تنصيب البرنامج بأقصر الطرق وبأقل تفاعل يذكر، بواسطة هذه الطريقة تستطيع تحديد المستلزمات لعملية التنصيب مثلاً الفريم ورك وكذلك يمكنك إرسال التحديثات لبرنامجك، تستطيع نشر برنامجك إلى سيرفر على الانترنت أو سيرفر محلي أو النشر على سيديهات، تستطيع الوصول إلى هذه الطريقة من طرق التنصيب بسهولة فقط اذهب إلى قائمة Build واختر Publish تستطيع تغيير إعدادات عملية التنصيب هذه بواسطة الذهاب إلى قائمة Project ثم اختيار Properties وتغيير الإعدادات التي تريدها.

أما بالنسبة للطريقة الثانية WindowsInstaller فيمكنك استخدامه بفتح مشروع جديد في الفيجوال 2008 ثم الذهاب إلى Other Projects Types ثم اختيار Setup and Deployment واختر من اليمين القالب الذي تريد وقم ببناء مشروع التنصيب.

إضافة أخيرة لبرنامجك:

لعمل إضافة صغيرة لبرنامجك (قد لمحت عنها سابقاً) افتح المشروع الذي قمت ببرمجته وتعلمته خلال هذا الفصل والمسمى Arqam ثم أضف الكلمة Randomize إلى حدث Load التابع للفورم ثم اضغط Enter. الكلمة Randomize معناها العشوائية وتستخدم ساعة النظام لتوليد أرقام عشوائية حقيقية حيث تحدد كلمة Randomize بداية عشوائية لتوليد الأرقام ليست نفس البداية التي يحددها البرنامج بدون Randomize وتختلف هذه البداية من وقت إلى أخر باختلاف ساعة النظام. بدون الكلمة Randomize يقوم البرنامج بتوليد أرقام عشوائية ولكن من بداية

معينة بحيث إذا تم تشغيل البرنامج أكثر من مرة وتم متابعة الأرقام العشوائية سنجد أنها متطابقة مع بعضها البعض. بعد إضافة هذا التحديث البسيط لبرنامجك الأول قم بحفظ المشروع.

خلاصة سريعة للفصل الثاني:

لعمل	**قم بالتالي**
إضافة بعض المكونات لواجهة المستخدم	اذهب إلى صندوق الأدوات واختر المكونات التي تحتاجها لواجهة المستخدم، بعد إضافة المكونات اللازمة قم بتغيير حجمها ومكانها على الفورم وغير مساحة الفورم لتتناسب مع المكونات والحاجة.
تحريك مكون معين	قم بتحديد المكون بواسطة الماوس ثم قم بعملية السحب إلى المكان الذي تريد على الفورم. أو قم بتغيير خصائص المكون المحدد.
تغيير حجم مكون	قم بتحديد المكون ستظهر ثمانية مربعات بيضاء صغيرة عليه، اضغط على احدها وقم بالسحب لتغيير حجم المكون، أو استخدم نافذة الخصائص.
حذف مكون	قم بتحديد المكون ثم اضغط Delete
فتح محرر الكود	قم بالضغط Double-Click على الفورم. أو قم بـــ Right-Click على الفورم واختر View Code.
حفظ المشروع	من قائمة File اختر Save All أو اضغط زر الحفظ 
تحويل المشروع إلى ملف exe	اذهب إلى قائمة Build اختر Build أو Rebuild.
إعداد المشروع	اذهب إلى قائمة Build واختر منها Publish واختر الخيارات الملائمة

للنشر وللتوزيع	لمشروعك للحصول على ملف التنصيب بطريقة الــ ClickOnce
فتح مشروع سابق.	من قائمة File اختر Open Project ثم اختر مشروعك لفتحه. أو من نفس القائمة اختر Recent Projects ثم اختر مشروعك. أو اختر مشروعك من قائمة أخر المشاريع الظاهرة في يسار بيئة التطوير تحت خانة Recent Projects

الفصل الثالث: التعامل مع الأدوات

الأدوات: هي تلك المكونات التي تضاف إلى واجهة المستخدم ليتفاعل معها المستخدم وقت تنفيذ البرنامج وتوجد في بيئة التطوير في صندوق الأدوات على اليسار من بيئة التطوير (بشكل افتراضي وتستطيع تغيير مكانه) بسهولة جدا نقوم بإضافة المكونات إلى الفورم بواسطة السحب والإلقاء بالماوس. الأدوات الموجودة في صندوق الأدوات مرتبة بحسب الفئة: فهناك الأدوات الشائعة الاستخدام تحت فئة Common Controls وهناك أدوات الاحتواء Containers وهناك أدوات الطباعة Printing (استخدمنا بعضاً من هذه الأدوات في الفصل السابق) سوف تعرف الكثير عن بقية الأدوات منها أدوات المستخدمة للتعامل مع قواعد البيانات وكذلك أدوات صفحات الإنترنت لاحقا في هذا الكتاب. في هذا الفصل سوف تتعلم كيفية تعرض معلومات معينة في صناديق النص TextBox وكذلك التعامل مع التاريخ والوقت في نظامك، استقبال مدخلات المستخدم، وفتح صفحة انترنت في بيئة التطوير. الأمثلة في هذا الفصل ستساعدك على فهم الخصائص والكود والكائنات.

الاستخدام المبسط للأدوات: برنامج The Hello World Program

في العرف البرمجي يوجد برنامج يسمى Hello World، إذا أردت دراسة أي من لغات البرمجة فلا بد من المرور بهذا البرنامج، هذا البرنامج عبارة عن شاشة أو صفحة انترنت مكتوب عليها Hello World. يبين هذا البرنامج مدى سهولة التعامل مع لغة البرمجة المعينة وكيفية البداية معها. في البرمجة المبنية على الحرف Character-Based Programming يأخذ برنامج Hello World سطرين برمجيين أو ثلاثة فقط ولكن مع تطور لغات البرمجة وتعقد أنظمة التشغيل أصبح برنامج Hello World أكثر تعقيداً مما سبق وقد يصل إلى عشرات الأسطر البرمجية. مع برنامج الفيجوال بيسك 2008 لحسن الحظ البرنامج سهل جدا فبالإمكان كتابة البرنامج بإضافة اثنين من المكونات واستخدام خاصيتين وسطر واحد من الكود فقط لذلك لابد من تجربة العملية.

١- افتح الفيجوال 2008 ثم اذهب إلى قائمة File واختر New Project، ستظهر لك نافذة تأكد من أن اللغة Visual Basic وان الفئة هي Windows ثم اختر Windows Forms Application قم بتغيير الاسم بالأسفل من WindowsApplication1 إلى MyHello



الأمثلة التي ستتعلمها خلال هذا الكتاب سيبدأ اسمها بـــ My تمييزاً لها عن تلك المرفقة مع هذا الكتاب، فاسم المثال المرفق Hello بينما الذي ستقوم بعمله اسمه MyHello.

بعد تغيير الاسم ستكون معك مثل هذه الشاشة:

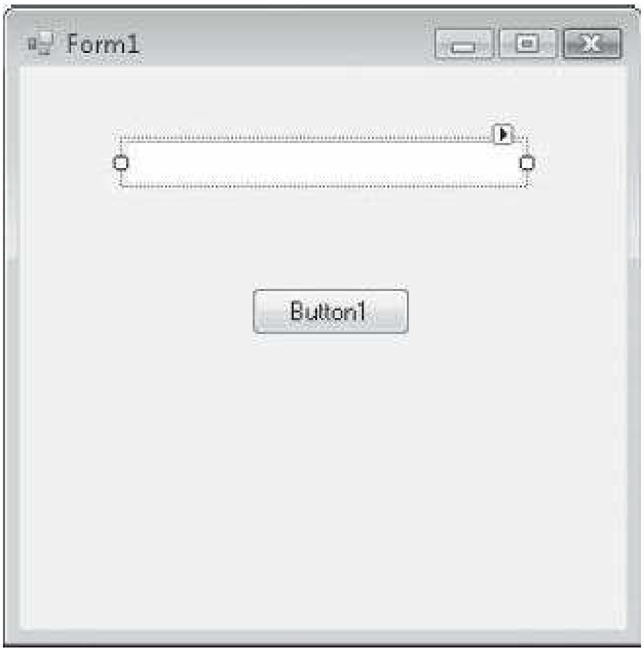


اضغط OK لإنشاء المشروع. بعد إنشاء المشروع سيظهر أمامك فورم فارغ. الأدوات التي سنستخدمها في هذا المشروع هي 1-زر Button، 2-صندوق للنص TextBox، تستخدم صناديق النص لإظهار النصوص أو لاستقبال النصوص المدخلة من قبل المستخدم خلال مرحلة تشغيل البرنامج. صندوق النص هنا لإظهار الجملة Hello World خلال مرحلة تنفيذ البرنامج بعد الضغط على الزر.



القراء الذين استخدموا النسخ السابقة من فيجوال بيسك سيلاحظون عدم وجود قيمة في الخاصية Text التابعة للـــ TextBox وإنما الخاصية فارغة.

بعد إضافة الزر وصندوق النص ستكون الفورم مثل الشكل التالي:



بقي لنا الآن أن نضيف الكود للزر Button1 في حدث Click هذا الكود

```
Textbox1.text = "Hello World"
```

ستلاحظ عند طباعتك للجملة أعلاه وعند وصولك النقطة بعد TextBox1 إن بيئة التطوير ستقوم بإظهار قائمة بالخيارات الممكن إدخالها للصندوق النص تجاهل القائمة وواصل الطباعة، فقط في حالة نسيت الخاصية التي تريدها أو تريد أن تبحث عن الإضافات الجديدة التي لا تعرفها يمكنك مشاهدة بنود القائمة. الجملة البرمجية أعلاه التي كتبتها تقوم بتغيير خاصية النص الموجود بداخل Textbox1 إلى النص Hello World سيتم تغيير الخاصية خلال مرحلة تنفيذ البرنامج.

قم بتشغيل البرنامج بالضغط على F5 بعد تشغيل البرنامج اضغط على زر OK الموجود في الفورم. سيقوم البرنامج بكتابة Hello World في مربع النص الموجود ضمن الفورم. قم بإغلاق البرنامج بالضغط على زر الإغلاق المتوفر في الفورم في أعلى يمين الفورم. بعدها قم بحفظ البرنامج الذي صممته (راجع الفصل الثاني لتعرف كيفية حفظ المشاريع).

ستجد نسخة من البرنامج موضوع في مجلد رقم 003 المرفق مع الترجمة. هذه صورة للبرنامج



تعرفت الآن على بعض أدوات بيئة التطوير وهما الزر Button و صندوق النص Textbox وتعرفت على بعض الخصائص الخاصة بهما، حان الوقت لتعرف أكثر عن بعض الأدوات الأخرى الخاصة ببيئة التطوير.

استخدام الأداة DateTimePicker

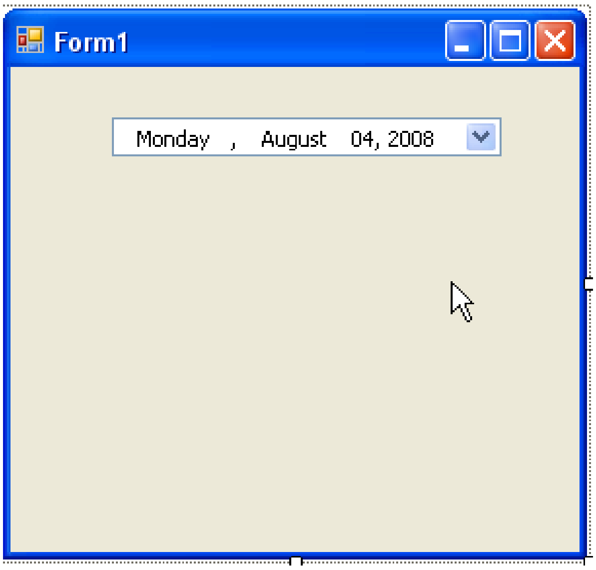
بعض أدوات الفيجوال بيسك 2008 تستخدم لعرض البيانات للمستخدم والبعض الآخر لأخذ البيانات من المستخدم، الأداة التي ستتعرف عنها الآن تستخدم لعرض تقويم مرئي يمكن للمستخدم من اختيار التاريخ والوقت، ستتعرف كيف وبسهولة يتم استقبال مدخلات المستخدم وكذلك كيف يتم معرف هذه المدخلات من خلال بيئة الفيجوال بيسك 2008.

برنامج تاريخ الميلاد:

برنامج تاريخ الميلاد هو عبارة عن برنامج يستخدم لاستقبال مدخلات المستخدم عن تاريخ ميلاده ثم تقوم بعرض التاريخ المدخل في صندوق حوار message box. نبدأ مع هذا البرنامج.

من قائمة File اختر New Project (بعض هذه الخطوات نقوم بتكرارها أكثر من مرة عندما نريد أن نبني مشروع جديد لذلك تعرف عليها ولا تنسها).

اختر اللغة Visual Basic ثم اختر Windows Forms Application وسمي مشروعك MyBirthday وستقوم بيئة التطوير بإنشاء المشروع وسيظهر لك فورم فارغ في بيئة التطوير، قم بإضافة الأداة DateTimePicker إلى وسط أعلى الفورم كما في الصورة التالية:



نلاحظ في الصورة أعلاه أن DateTimePicker يقوم بإظهار التاريخ ويوم الأسبوع لليوم الذي نشاهد فيه الفورم في الصورة أعلاه يوم الاثنين الرابع من شهر أغسطس لعام 2008 نستطيع تعديل عملية إظهار التاريخ والوقت بتغيير الخاصية Value التابعة لـ DateTimePicker حيث تستطيع إظهار تاريخ معين في كل مرة يتم فيها تشغيل البرنامج. قم بإضافة زر Button إلى الفورم ليقوم تاريخ الميلاد بعد أن يتم اختياره من الـ DateTimePicker. في خاصية الـ Text التابعة للزر Button1 نكتب التالي "أظهر تاريخ ميلادي". الآن نقوم بإضافة كود للزر Button1 في حدث Click(حدث الضغط على الزر)، طبعاً نستطيع إضافة الكود عند أي حدث مثلاً عند حدث MouseHover (تحرك الماوس فوق الزر) أو MouseLeave (مغادرة الماوس من فوق الزر). وبما إننا نريد أن نضغط فوق الزر ليقوم البرنامج بإظهار التاريخ الذي تم اختياره من الـ DateTimePicker فعلينا إذن أن نختار الحدث Click التابع للزر Button1 ونكتب فيه هذا الكود:

```
        MsgBox("تاريخ ميلادك هو       " & DateTimePicker1.Text)
        MsgBox("رقم اليوم في سنة ميلادك هي   " & _
(DateTimePicker1.Value.DayOfYear.ToString
```

الكود أعلاه يقوم بعرض اثنين صناديق حوار بعد الضغط على الزر صندوق الحوار الأول يعرض لك تاريخ ميلادك والذي قد اخترته من الــ DateTimePicker والصندوق الحواري الثاني يعرض لك رقم اليوم من السنة (فالأول من يناير يعتبر اليوم الأول من السنة والعاشر من فبراير يعتبر اليوم الأربعون من السنة). لاحظ فقط في السطر الثاني فبسبب طول السطر تم الانتقال إلى السطر الثالث فقط بإضافة العلامة _ قبل الكلمة التي تريد أن تنقلها إلى السطر الثاني ثم تضغط Enter فينتقل بقية السطر إلى الذي يليه.

> السطر البرمجي قد يحتوي على 65000 حرف أو أكثر لكن يفضل إذا كانت عدد الحروف لا تزيد عن 80 في السطر الواحد لكي يسهل مراجعتها وقراءتها. يمكنك الانتقال إلى السطر الذي يليه بإضافة العلامة _ قبل الكلمة التي تريد أن تنقلها إلى السطر الذي يليه. (لا يمكنك استخدام هذه العلامة _ للانتقال إلى السطر الذي يليه بداخل الجمل المغلقة بعلامات التنصيص " "

مشروع تاريخ الميلاد الذي قمت بتصميمه يوجد منه نسخة في الملفات المرفقة في الملف رقم 004 تستطيع فتحة والمقارنة بينه وبين الذي قمت بتصميمه. بعد تشغيل البرنامج واختيار تاريخ ميلادك من DateTimePicker ثم الضغط على الزر "أظهر تاريخ ميلادي" سيظهر أمامك صندوق حوار تاريخ ميلادك هو وسيظهر لك التاريخ الذي حددته، اضغط موافق أو OK سيظهر لك صندوق حوار آخر ليوضح لك رقم اليوم التي ولدت فيها من السنة. جرب تشغيل البرنامج من اجل أن تشاهد هذه العملية وكيفية التعامل مع هذه الأداة.

> لتجعل DateTimePicker يقوم بإظهار التاريخ فقط أو الوقت قم بتغيير الخاصية Format التابعة له إلى Time

ملاحظات حول المصطلحات:

استخدمنا العديد من المصطلحات في هذا الكتاب حتى الآن وهناك العديدون الذين لا يعرفون هذه المصطلحات بشكل تام، وعليه وجب علينا الآن أن نوضح بعضاً من هذه المصطلحات.

الجمل البرمجية(Program Statements):

الجمل البرمجية هي عبارة عن الجمل المكتوبة في السطور البرمجية (خانة الكود) وتقوم هذه الجمل بعمل ما خلال مرحلة تنفيذ البرنامج لان الكومبايلر (المترجم إلى لغة الآلة) يقوم بقراءة هذه الجمل وتنفيذها. يختلف طول هذه الجمل بحسب الحاجة فبعضها قد يكون طويلاً والبعض الآخر قد يحتوي على كلمة واحدة لكن جميعها يجب أن تتبع طريقة الـــ syntax rules القواعد البرمجية أو الطرق البرمجية التي يتقبلها المترجم أو الكومبايلر. في فيجوال بيسك 2008 الجمل البرمجية قد تحتوي على كلمات مجوزة، خصائص، أسماء كائنات، متغيرات، أرقام، رموز، وقيم أخرى انظر الفصل الثاني وكذلك الفصل الخامس.

الكلمات المحجوزة (Keywords):

الكلمات المحجوزة keywords هي كلمات محجوزة في بيئة التطوير هذه الكلمات تتعامل مع الكومبايلر بالطريقة التي قد حددت سلفاً من قبل مطوري لغة البرمجة (مايكروسوفت). مثل الكلمة end وتستخدم لإغلاق البرنامج أو التطبيق وعليه فلا يمكنك أن تقوم بتعريف متغير بنفس الكلمة. الكلمات المحجوزة تعتبر جزء من بنية الجمل البرمجية التابعة للفيجوال بيسك معظم الكلمات المحجوزة تظهر باللون الأزرق في محرر الكود.

المتغيرات (Variables):

المتغيرات هي عبارة عن حاضنات أو حافظات للبيانات تحفظ البيانات بشكل مؤقت ويتم تعريف المتغيرات باستخدام كلمة Dim قبل المتغير وتقوم هذه المتغيرات بحفظ البيانات بشكل مؤقت

وعادة ما تكون هذه البيانات عبارة عن أسماء ملفات، أرقام، تواريخ، صور، بيانات ثنائية (الصفر والواحد)، خصائص وقيم أخرى.

الأدوات (Controls):

الأدوات هي عبارة عن تلك الأدوات التي تقوم بإضافتها إلى الفورم مثل الأزرار، صناديق النص الليبلات وصناديق الصور وغير من الأدوات التي يمكنك إضافتها إلى الفورم (عادة ما تكون في واجهة المستخدم). اقرأ الفصول من 2 إلى 4.

الكائنات (Objects):

الكائنات هي عبارة عن عناصر التي تقوم بصنعها بواسطة برنامج الفيجوال بيسك باستخدام أحد الأدوات الموجودة في صندوق الأدوات Toolbox. في نفس الوقت الكائنات قد تكون مقدمة من ملحقات نظام التشغيل والعديد من هذه الكائنات تحتوي على بيانات. للتوضيح أكثر نقول إن الفورم واحد من هذه الكائنات صندوق الحوار كذلك. بعبارة أخرى الفورم هو عبارة عن فئة class (كلاس سنقوم بتعريفها لاحقاً وبصراحة لم أجد مصطلح مطابق لها 100% لذلك سنستخدم مصطلح كلاس) التي تدعم الخصائص والطرق Methods والأحداث Events. تمتلك الكائنات خاصية الوراثة (الوراثة هي مجموعة من عمليات نظام التشغيل التي تقوم بعملها الكائنات بدون برمجة مسبقة من قبلك لان البرمجة موجودة مسبقاً في النظام الأب نظام التشغيل لذلك فإننا نسمي هذه الخاصية الوراثة فمثلا الفورم يعرف كيف تكبره وتصغره بدون برمجة مسبقة لهذا الأمر.) (اقرأ الفصول من 1 إلى 4).

الكلاس (Class):

الكلاس أو القالب عبارة عن قالب لكائن أو أكثر والذي يحدد فيه ماذا يفعل هذا الكائن. وعليه فالكلاس أو القالب يحدد ماذا يجب أن يفعل الكائن وليس الكائن نفسه. في الفيجوال بيسك 2008

تستطيع استخدام القوالب الموجودة ضمن بيئة التطوير مثل System.Math أو System.Windows.Forms.Form لبناء قوالبك الخاصة ووراثة الخصائص والطرق Methods والأحداث من القوالب المذكورة (الوراثة تسمح للقالب الذي تنشئة بأخذ الخصائص وطريقة التعامل من قالب Class أخر موجود مسبقاً). في الفصلين **5** و **16** سوف تعرف أكثر عن الكلاس.

مجالات الأسماء (Namespaces) :

مجالات الأسماء (فضاءات الأسماء) عبارة عن مكتبات أو قوالب **Classes** مرتبة تحت اسم معين مثل System.Windows أو System.Diagnostics لتصل إلى هذه الفئات (Classes) لابد أن تكتب **Imports** في أعلى الفورم متبوعا باسم مجال الأسماء المحدد. للمزيد انظر الفصل الخامس.

الخصائص (Property)

الخاصية هي عبارة عن قيمة معينة محمولة بواسطة كائن معين. فمثلاً الزر **Button** لدية خاصية النص Text هذه الخاصية تحدد ما هو النص المكتوب في الزر. وكذلك الخاصية **Image** والتي تحدد مسار الصورة التي على الزر. في فيجوال بيسك تستطيع من تغيير الخصائص وقت التصميم بسهولة من نافذة الخصائص وكذلك يمكنك تغيير الخصائص وقت تنفيذ البرنامج بواسطة الكود، القاعدة العامة لتغيير الخاصية كالتالي:

Object.Property = Value

وهي: الكائن.الخاصية = قيمة الخاصية

فمثلاً يمكن تغيير خاصية النص في الكائن الزر كالتالي:

"Button1.Text = "Hello

فالكائن هو Button1 والذي هو الزر والخاصية هي Text والتي هي النص والقيمة هي النص .Hello

الأحداث (Event Procedure):

الأحداث هي عبارة عن كود معين يتم تنفيذه عندما تتم معالجة كائن ما في البرنامج فإذا لدينا الزر Button1 فالحدث Click يحدد ماذا يقوم البرنامج بتنفيذه في حالة النقر على الزر. تستطيع كتابة أوامر معينة لتغيير بعض الخصائص في البرنامج وكذلك لتنفيذ أوامر معينة في البرنامج.

الطرق (Methods):

الطرق Methods هي عبارة عن أوامر برمجية معينة لتقوم ببعض الأعمال أو تنفذ خدمات معينة لكائن معين في داخل البرنامج. الكود سيكون كالتالي:

(Object.Method(Value

كائن.طريقة(قيمة للطريقة) كالمثال التالي:

("ListBox1.Items.Add("Books

في الكود أعلاه استخدمنا الطريقة Add ومعناها أضف (فعل أمر) كلمة Books إلى البنود الخاصة بالكائن ListBox1 (القائمة).

يوجد بعض التقارب بين الخصائص Properties والطرق Methods ولكننا نستطيع أن نفرق بينهما بواسطة مكانهما في الكود البرمجي. و الأمثلة أعلاه توضح ما تكلمنا عنه وسوف تعرف أكثر عن الطرق والخصائص خلال الفصول من الأول إلى الخامس.

أدوات لجلب المدخلات من المستخدم

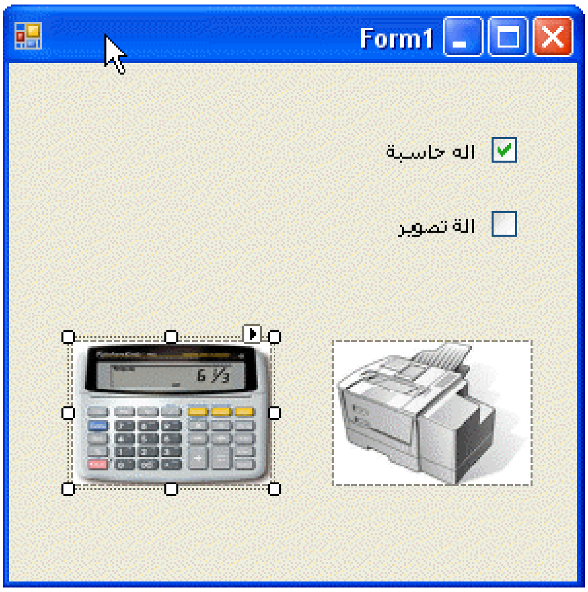
تقدم بيئة الفيجوال بيسك العديد من الأدوات لجلب المدخلات من مستخدم البرنامج صناديق النص TextBox، القوائم Menus تقوم باستقبال المدخلات بواسطة النقر عليها بالماوس أو باختيارها بواسطة الكيبورد. صناديق الحوار وسيلة من وسائل اخذ المدخلات من المستخدم هناك العديد من الأدوات الأخرى التي سوف نعرف عنها لاحقاً منها ،ListBox ،CheckBox ،RadioButton ComboBox. هذه الأدوات وغيرها من أدوات بيئة التطوير تقوم بأخذ المدخلات من المستخدم فمثلاً الـــ RadioButton قد يوكل إليها معرفة ما إذا كان الشخص ذكراً أو أنثى. أما الـــ CheckBox فنستخدمها في حالة إذا ما أردنا من المستخدم اختيار أكثر من خيار فمثلا نريد من المستخدم اختيار الأحجار الكريمة التي يحبها بعض المستخدمين سيختار العقيق والزمرد والبعض الأخر سيختار العقيق واللؤلؤ والبعض الآخر سيختار اللؤلؤ والمرجان والزبرجد. الأداة ListBox تتيح لك اختيار أكثر من خيار بطريقة التظليل واستخدام الزر Ctrl. الـــ ComboBox يتيح لك اختيار واحد ولكن من قائمة منسدلة من الخيارات والاقتراحات. سوف تتعرف أكثر على هذه المكونات والأدوات في المستقبل وسنقوم بتصميم برنامج يحتوى على معظمها. الآن سنقوم بتصميم برنامج يحتوي على المكون CheckBox:

قم بإنشاء مشروع جديد وقم بتسميته MyCheckBox، بعد ظهور الفورم قم بإضافة المكون CheckBox إلى الفورم (قم بإضافة اثنين من الـــ CheckBox) واجعل احدهما فوق الآخر ثم أضف اثنين صناديق للصور PictureBox، وبعد ذلك قم بتعديل بعض الخصائص للمكونات التالية كالتالي:

الكائن	الخاصية	التعديل اللازم
Form1	RightToLift	Yes
الخاصية أعلاه تجعل الفورم يظهر من اليمين إلى اليسار، وتذكر دائماً إن قابلية الوراثة الموجودة ضمن بيئة الفيجوال بيسك تنقل لك بعض الخصائص (مثل الخاصية أعلاه من الفورم إلي بقية		

المكونات المتواجدة ضمن الفورم) وعليه فجميع المكونات التي ستكون داخل الفورم سترث الخاصية من اليمين إلى اليسار ولا نحتاج لتغيير هذه الخاصية لكل مكون على حده. تحويل اتجاه الفورم من اليمين إلى اليسار يفيدنا في حالة برمجة التطبيقات ذات الواجهة العربية، حيث قد ألف المستخدمون العرب على البرامج ذات الواجهة العربية.		
CheckBox1	Text	اله حاسبة
CheckBox1	Checked	True
CheckBox2	Text	آلة تصوير
PictureBox1	SizeMode	StretchImage
PictureBox1	Image	اضغط على الزر واختر صورة الآلة الحاسبة الموجودة في المرفقات ملف 005
PictureBox2	SizeMode	StretchImage
PictureBox2	Image	اضغط على الزر واختر صورة آلة التصوير الموجودة في المرفقات ملف 005

بعد إضافة المكونات وتعديل الخصائص المطلوبة سيظهر لك الفورم بالشكل التالي:



الآن هل تستطيع أن تعرف ماذا نريد من البرنامج، نريد من البرنامج في حالة تأشير خانة الآلة الحاسبة إظهار الآلة الحاسبة وفي حالة تأشير آلة التصوير إظهار آلة التصوير. طبعا في الحالة الطبيعية (أي بعد تشغيل البرنامج سيتكون خانة الآلة الحاسبة مؤشرة وعلية فستكون صورة الآلة الحاسبة ظاهرة أما خانة آلة التصوير (تذكر أننا جعلناها مؤشرة في الحالة العادية عند تعديل خصائص المكونات) فلن تكون مؤشرة وعلية فصورة آلة التصوير لن تكون ظاهره). عرفنا كيف سيكون البرنامج في الحالة العادية ولكن كيف سيكون في حالة تأشير خانة الآلة الحاسبة أو خانة آلة التصوير هذا لابد أن نجيب عنه في مرحلة الكود: (ستلاحظ تكرار بعض التعليمات أكثر من مرة بين هذا الفصل والفصل السابق لكن في حالة إذا ما كنت مبتدأ فهذا سيعطيك معلومات إضافية)

مرحلة الكود:

اضغط Double-Click على CheckBox1 واكتب الكود التالي(ستلاحظ إن بيئة التطوير تنقلك إلى الحدث CheckedChanged التابع لـ CheckBox1):

```
'  تعني ان الخانة غير مؤشرة0 تعني ان الخانة مؤشرة القيمة1طبعا القيمة
        ' الكود في الاسفل يقول اذا كانة الخانة مؤشرة فقم بالتالي
        If CheckBox1.CheckState = 1 Then
            ' هذه هي الاوامر التي ينفذها البرنامج في حالة صحة الشرط اعلاه
```

```
            ' قم بـإظهـار الـصورة المـوجودة ضمن المسار المحدد
            ' مسار الـصورة قـد يختلف في جهـازك عن المسار الموجود
            ' في جهـازي لـذلـك تـنبـة وقم بـكتـابـة المسار الـصحيح لـلصورة
            PictureBox1.Image = System.Drawing.Image.FromFile _
            ("C:\Documents and
Settings\Administrator\Desktop\vbsbs\005\calc.jpg")
            'الامـر الـسابـق كـان لـتحميل الـصورة في صنـدوق الـصورة
            ' امـا الامـر الـتـالي فـهـو لاظهـار الـصورة المحملة في صنـدوق الـصورة لانها قـد
تـكون مخفيـة
            PictureBox1.Visible = True
            ' تـنبـه لـلامـر المـوجود في السطر الـتـالي فـهـو احد اهـم الاوامـر الـشرطيـة
            ' الامـر الـتـالي معنـاه كـل شي غير مـاسبق
            ' اي اذا صادفـتك أي أي أي أي أي حالـة لـيست كـالحـالـة او كـاحـالات الـسابـقة
فـقم بـالـتـالي
        Else
            'إخفـاء الـصورة في صنـدوق الـصورة
            PictureBox1.Visible = False
        End If
    End Sub

    Private Sub CheckBox2_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles CheckBox2.CheckedChanged
        If CheckBox2.CheckState = 1 Then
            PictureBox2.Image = System.Drawing.Image.FromFile _
            ("C:\Documents and
Settings\Administrator\Desktop\vbsbs\005\copier.jpg")
            PictureBox2.Visible = True
        Else
            PictureBox2.Visible = False
        End If
```

انظر توجد سطور خضراء ضمن الكود هذه السطور للتذكير فقط ولا تدخل ضمن عملية معالجة الكود، والدليل على ذلك وجود الحرف ' في أول السطر. ثم ننقر Double-Click على الـــ CheckBox2 ونكتب الكود التالي:

```
If CheckBox2.CheckState = 1 Then
            ' تـأكـد من المسار الـتـابـع لـلصورة في جهـازك وقم بـتغييره اذا لـزم الامـر
            PictureBox2.Image = System.Drawing.Image.FromFile _
            ("C:\Documents and
Settings\Administrator\Desktop\vbsbs\005\copier.jpg")
            PictureBox2.Visible = True
        Else
            PictureBox2.Visible = False
        End If
```

قم بالنقر Double-Click على الفورم سينقلك مباشرة إلى الحدث Load التابع للفورم قم بكتابة الكود التالي لإخفاء صورة آلة التصوير في الحالة الطبيعية (لحظة تشغيل البرنامج):

```
PictureBox2.Visible = False
```

قم بتشغيل البرنامج بالضغط على F5 وقم بتأشير خانة الآلة الحاسبة والعكس ولاحظ ظهور واختفاء الآلة الحاسبة.

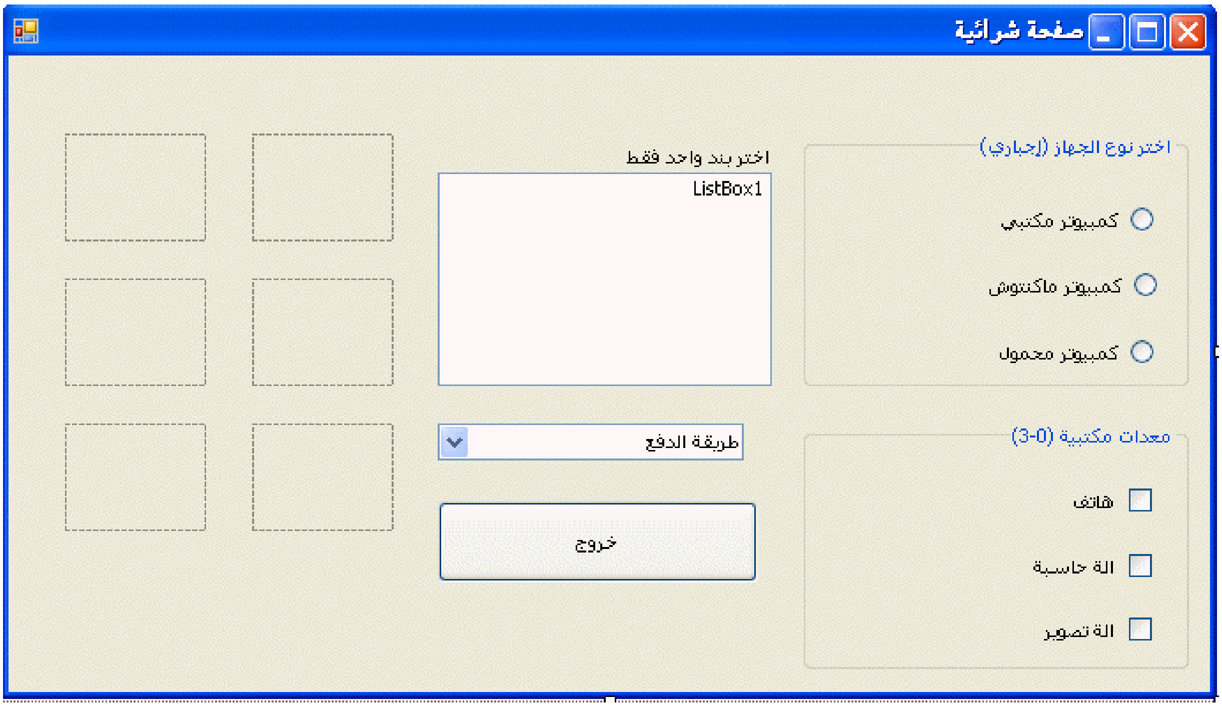
ستجد نسخة من المشروع أعلاه مع المرفقات في المجلد رقم 005

في المثال أعلاه تعرفت على احد أنواع استقبال المدخلات من المستخدم فعندما يقوم المستخدم بتأشير الخانات فهو يقوم في ذلك الوقت بإرسال مدخلات إلى برنامجك، هل تتذكر يوم تسجيك في احد المنتديات حينما طلب منك المنتدى تحديد ما إذا كنت ترغب باستقبال رسائل من الأعضاء في المنتدى، تأشيرك لذلك المربع يعني قبولك استقبال رسائل من أعضاء المنتدى ويعتبر نوع من أنواع الحصول على المدخلات من المستخدم.

برنامج المدخلات التجريبي:

لديك الآن خبرة لا باس بها في الـــ CheckBox وكذلك في الأحداث والخصائص والمكونات.
الآن سأعطيك برنامج مدخلات تجريبي لترى كيف يمكن الاستفادة من بعض المكونات في معرفة مدخلات المستخدم.

البرنامج في المرفقات الملف رقم 006 بعد فتح البرنامج ستجد البرنامج كما في الشاشة التالية:



شَغل البرنامج بالضغط على F5 وجرب الخيارات الموجودة في المشروع مثلاً: كمبيوتر مكتبي أو محمول ثم اختر هاتف وآلة حاسبة مع آلة تصوير ثم لاحظ التغيرات على الفورم.

مرحلة الكود:

بما أننا قد تعرفنا سابقاً على العديد من مراحل الكود وطريقة كتابة الأكواد ولم كتبنا الكود بهذه الطريقة (ستلاحظ أننا سنقلص من الشرح الممل للكود). قمنا باختصار بعض الكود ببعض الطرق في المثال أعلاه، لاحظ أن هناك عدد ثلاثة لنوع الجهاز وفي كل مرة تختار نوع الجهاز تظهر صورة للجهاز المختار سواء كان محمول أو مكتبي أو ماكنتوش. أما الـــ ListBox1 فيحتوي على أربعة بنود مع ثلاثة خيارات من المعدات المكتبية وخيارين للدفع بينما يوجد فقط ستة مربعات للصورة فكيف تم ذلك، لاحظ أن اختيار نوع الجهاز واختيار احد البنود في قائمة ListBox1 أو اختيار طريقة الدفع يسمح لك باختيار بند واحد فقط ولا يسمح لك باختيار أكثر من خيار وعلية فقد تم وضع مربع صورة واحد لكل قائمة من القوائم أعلاه وعليه فقد وفرنا العديد من صناديق الصور والعشرات من الأسطر في الكود. فقط المعدات المكتبية تم وضع مربع صورة

لكل خيار بسبب أن الخيار مفتوح لاختيار أكثر من بند. قم بعمل Right-Click على الفورم في المشروع المرفق رقم 006 ثم اختر View Code ثم لاحظ الكود أولاً ثم ضع الكود التالي تحت الحدث Load التابع للفورم ليتم تنفيذ الكود أول ما يفتح البرنامج:

```
    RadioButton1.Checked = True
        ListBox1.Items.AddRange(New Object() {"هارد خارجي", "طابعة", "ماسح ضوئي", "سماعات"})
```

السطر الأول من الكود ليتم اختيار RadioButton1 (التابعة للكمبيوتر المكتبي) في الحالة الطبيعية (أول ما يفتح البرنامج). السطر الثاني من الكود يقوم بتعبئة الـــ ListBox1 بالبنود:

هارد خارجي، طابعة، ماسح ضوئي، سماعات.

الكود التالي تابع RadioButton1 تحت الحدث CheckedChanged:

```
Pc.Visible = True
        Pc.Image = Global.Input.My.Resources.Resources.PComputr
```

تم إظهار مربع الصورة وتحميل صورة الكمبيوتر له من ملف Resources المرفق مع المشروع. أما الكود الذي يليه فهو تابع للـــ RadioButton2 ويقوم بتحميل الصورة التابعة للماكنتوش أما الذي بعده فيحمل صورة الكمبيوتر المحمول لاحظ أننا لم نخفي الصورة الأولى الموجودة في مربع الصورة لأننا سنقوم بتحميل الصورة في نفس مربع الصورة وليس في مربع صورة آخر.

الكود التالي يتبع المكون CheckBox1 تحت الحدث CheckedChanged:

```
If CheckBox1.Checked = -1 Then
            Tel.Image = Global.Input.My.Resources.AnswMach
            Tel.Visible = True
        Else
            Tel.Visible = False
        End If
```

لتوضيح الكود أعلاه انظر الصورة التالية:



ستجد إن المربع إما أن يكون مختار أو ليس مختار وهذا ما يسمى بالبرمجة بالخيار Boolean ومعناه صحيح أو خطأ. صحيح أو خطأ (صحيح = سالب واحد، خطأ = صفر)،للمزيد حول True صحيح أو False خطأ على هذا الرابط: (لـــ حامد الهادي عيد)

http://vb4arab.com/vb/showthread.php?p=15891#post15891

ففي الكود أعلاه طلبنا من الكمبيوتر التأكد إذا كان CheckBox1 مؤشر فإذا كان مؤشر يحمل صورة الهاتف وفي حالة العكس يقوم بإخفاء الصورة واستخدمنا نفس الكود مع CheckBox2 و CheckBox3. الكود الذي يليهم للتعامل مع البيانات في القائمة ListBox1 وبما أننا أمام أكثر من خيارين لم نستعمل أداة الشرط If ولكننا استخدمنا أداة شرط جديدة هي Select Case، نستخدم Select Case إذا كنا أمام أكثر من ثلاثة خيارات، وفي العديد من الظروف سنعرفها في الفصول اللاحقة. لنراجع بنود القائمة في وضع التصميم نلاحظ القائمة فارغة ولكن بعد تشغيل البرنامج نلاحظ وجود أربعة بنود عليها من أين أتوا، من الكود الذي وضعناه في الحدث Load التابع للفورم وهو:

```
ListBox1.Items.AddRange(New Object() {"هارد خارجي", "طابعة", "ماسح ضوئي", "ساعات"})
```

ففي حالة اختيار احد البنود أعلاه ستظهر صورته في مربع الصورة المسمى LstIndex (سميناه بهذا الاسم ليسهل علينا تذكره). للعلم قائمة الـــ ListBox تقوم بترتيب البنود فيها من الرقم صفر وليس من الرقم واحد كما يظن البعض، لذلك فإذا أردنا اختيار البند الأول فإننا نختار البند رقم

صفر، وإذا أردنا اختيار البند الثاني فإننا نختار البند رقم واحد وهكذا. دعونا نتأمل في الكود التالي:

```
Select Case ListBox1.SelectedIndex

            Case 0
                LstIndex.Image = Global.Input.My.Resources.Harddisk
                LstIndex.Visible = True
            Case 1
                LstIndex.Image = Global.Input.My.Resources.Printer
                LstIndex.Visible = True

            Case 2
                LstIndex.Image = Global.Input.My.Resources.scanner
                LstIndex.Visible = True

            Case 3
                LstIndex.Image = Global.Input.My.Resources.Headset
                LstIndex.Visible = True

        End Select
```

السطر الأول يدعو الكمبيوتر لاختيار عملية الشرط بمعلومية الترتيب التابع للمكون ListBox1 فإذا كان البند المختار هو البند الأول فسينفذ الكمبيوتر الأوامر الموجود تحت كلمة Case 0 وقبل كلمة Case 1 وهي إظهار الصورة التي تحددها له في مربع الصورة المحدد وإظهار الصورة في مربع الصورة. الكود الذي يليه تابع للمكون ComboBox1 تحت الحدث SelectedIndexChanged والذي يعني (تغير البند المختار في الكمبوبوكس) نلاحظ إن فكرة الكود التابع الكمبوبوكس نفس الفكرة السابقة في كود القائمة ListBox. أما الكلمة End في الكود فقد تعرفنا عليها سابقا في هذا الكتاب (اختبر معلوماتك!!!).

ستجد نسخة من المشروع أعلاه مع المرفقات في المجلد رقم 006
ستلاحظ أعلاه كيف تم استدعاء الصور للعرض بغير الطريقة التي تعلمتها مع التمارين السابقة (حيث تم استدعاء الصور من ملف Project Resources file الموجود ضمن ملفات المشروع باسم Resources التي يتم تحديدها من الخصائص– الخاصية

Image– ثم Project Resources file ثم Import ثم نختار الصورة التي نريد لتقوم بيئة التطوير بنقلها إلى ملف Resources ضمن ملفات المشروع)، في الحقيقة تستطيع استعراض الصور في برنامجك بأكثر من طريقة حسب الحاجة وكذلك حسب حجم البرنامج ونوعه (لان بعض البرامج تحتاج للكثير من الصور التي لا يمكن ذكر مسار كل صورة على حده لذلك فهناك العديد من الطرق للاحتفاظ بالصور ضمن ملفات البرنامج، ولان بعض الصور لا تريد لها أن تُنسخ إلى مكان أخر في الكمبيوتر مثل الشعارات وغيرها من الصور). لذلك وفرت لنا بيئة التطوير دمج الصور ضمن المشروع أو التطبيق للمزيد اذهب إلى التعليمات وابحث عن هذين الموضوعين: (1- How to: Create Embedded Resources), (2-Accessing (Application Resources

استخدام المكون: LinkLabel

إذا صممت تطبيق أو برنامج ما وتريد أن تضع عنوان لصفحتك في أي فورم، كيف يتم ذلك، لا تذهب بخيالك وتقول لي استخدم الليبل Label، نعم تستطيع استخدام الليبل Label ولكن الليبل فقط لعرض النصوص لكنك تريد وصلة إلى موقعك على الانترنت بمجرد أن يضغط المستخدم عليها تنقله إلى صفحتك على الانترنت باستخدام المتصفح الافتراضي في نظام التشغيل. ممكن تقوم بتصميم برنامج يعمل على سيرفر على الانترنت وكيف سيقوم المستخدم بالتنقل، احتمال تقوم بتصميم متصفح للانترنت أو تريد أن تفتح صفحة من صفحات الانترنت ضمن برنامج بالاعتماد على قدرات الانترنت إكسبلورر IE إذاً لابد من استخدام LinkLabel لهذا الغرض.

قم بإنشاء مشروع جديد وتسميته بــ MyWebLink بعد فتح الفورم ضمن بيئة التطوير قم بإضافة المكون LinkLabel إلى الفورم قم بتغيير الخاصية Text إلى "منتدى فيجوال بيسك للعرب" والخاصية LinkVisited إلى True لتغيير لون النص في حالة زيارة الموقع على الانترنت. قم بالنقر على الفورم واختر الخاصية Text التابعة للفورم وقم بتغييرها إلى: "فتح صفحة انترنت" قم

بتغيير الخاصية RightToLift إلى Yes (الشخص المتابع من البداية يعرف لماذا هذه التغييرات) سيكون مشروعك بهذا الشكل:



قم بعمل Double-Click على المكون LinkLabel ستنتقل إلى منطقة الكود تحت الحدث LinkClicked نقوم باستخدام طريقة Method (هل تتذكر Method!!!) لفتح صفحة انترنت باستخدام المتصفح الافتراضي في النظام (والذي غالباً ما يكون الـــ Internet Explorer أو Opera أو FireFox)، كما في هذا الكود:

```
System.Diagnostics.Process.Start("http://vb4arab.com/vb/")
```

ماذا إذا رأينا تشغيل الوصلة التي نريد في Internet Explorer بغض النظر عن المتصفح الافتراضي، بعبارة أخرى نريد أن نشغل وصلة المنتدى أعلاه في الـــ IE حتى إذا كان المتصفح الافتراضي هو الفايرفوكس. وعلية نستخدم الطريقة Process (Method) ولكن بطريقة مختلفة كهذا الكود:

```
System.Diagnostics.Process.Start("IExplore.exe", "http://vb4arab.com/vb/")
```

أو نكتب اسم البرنامج بين علامات التنصيص "" ثم علامة الفاصلة ، ثم موقع الانترنت. هذه الطريقة نستطيع استخدامها مع كل البرامج فمثلاً إذا كان لدينا ملف اسمه Textfile.txt بداخل القرص C:\ نريد أن نفتحه ببرنامج Word فنستخدم الكود:

```
System.Diagnostics.Process.Start("Winword.exe", "c:\Textfile.txt" )
```

وعليه فالطريقة Process مفيدة جداً لفتح الملفات والبرامج.

قم بتشغيل البرنامج أعلاه وشاهد كيف يتم فتح صفحة الانترنت، وكيف يتم تغيير لون الوصلة في حالة فتح الصفحة من قبل.

ستجد نسخة من المشروع أعلاه مع المرفقات في المجلد رقم 007

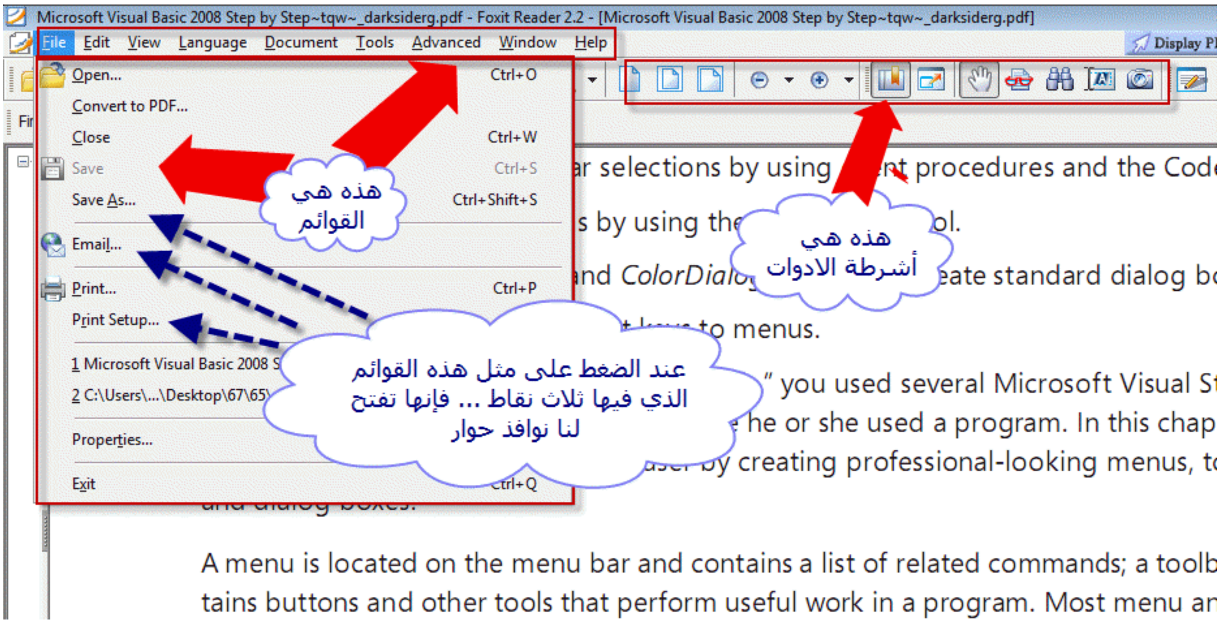
خلاصة الفصل الثالث:

إذا أردت أن	**قم بالتالي**
إضافة صندوق للنص	أضف المكون TextBox إلى الفورم
إضافة زر	أضف المكون Button إلى الفورم
تغير خاصية وقت تشغيل البرنامج	نستخدم الكود مثل : TextBox1.Text = "Hello"
إضافة زر الراديو	أضف المكون RadioButton إلى الفورم، في حالة هناك أكثر من زر راديو فيجب تجميع الأزرار ذات الهدف الواحد في المكون GroupBox.
إضافة زر تأشير	أضف المكون CheckBox إلى الفورم
إنشاء قائمة ListBox	أضف المكون ListBox إلى الفورم.
إنشاء قائمة منسدلة	أضف المكون ComboBox إلى الفورم.
إضافة بنود للقائمة ListBox	استخدام الطريقة Add تحت الحدث Load التابع للفورم أو تحت أي حدث مناسب كالتالي:ListBox1.Items.Add("Printer")

تدوين ملاحظة تذكيرية في منطقة الكود	اكتب هذه العلامة ' قبل الجملة التذكيرية التي تريد أن تكتبها في منطقة الكود كما المثال التالي: ' هذه جملة تذكيرية
فتح صفحة انترنت	نضيف المكون LinkLabel الفورم ثم نستخدم الطريقة Process.Start في منطقة الكود التابع للمكون LinkLabel

الفصل الرابع:التعامل مع القوائم Menu وصناديق الأدوات Toolbars وصناديق الحوار Dialog Boxes.

في البداية نريد أن نتذكر أننا درسنا في الفصل السابق كيفية التعامل مع المدخلات من قبل المستخدم سنبدأ في هذا الفصل بتعلم القوائم وأشرطة الأدوات وصناديق الحوار (التعامل) لنعرف ما هي القوائم والأشرطة ونوافذ الحوار نستعين بالصورة التالية من برنامج الـــ Acrobat برنامج مستعرض الكتب:

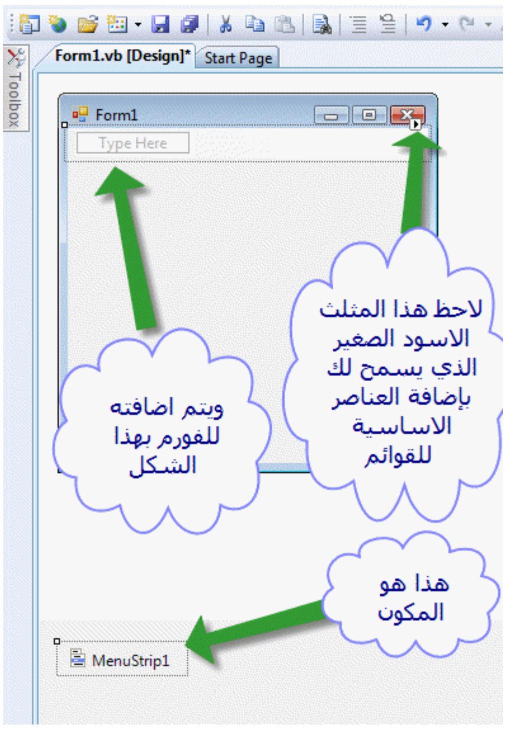


المكونين الذين سنتعلمهما في هذا الفصل هما القوائم MenuStrip، وأشرطة الأدوات ToolStrip. القوائم وأشرطة الأدوات تضيف منظر جمالي واحترافي للبرامج التي تقوم بتصميمها.

إضافة القوائم باستخدام المكون MenuStrip

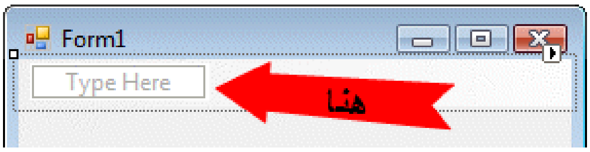
باستخدام المكون MenuStrip يمكنك إضافة القوائم إلى برنامجك وكذلك يمكنك التعديل على القوائم الحالية إضافة اللمسات الخاصة على القوائم مثل مفاتيح الاختصار وعلامات تأشير ☑ تستطيع إضافة القوائم الأساسية لبرنامجك بضغطة زر بدون تعقيدات. طبعاً بعد إضافة القوائم لبرنامجك لابد من إضافة الكود لهذه القوائم لان بيئة التطوير تساعدك فقط في تصميم القوائم،

افتح مشروع جديد وقم بتسميته MyMenu بعد ظهور الفورم اسحب المكون MenuStrip إلى الفورم. طبعا بيئة التطوير لن تقوم بوضع المكون فوق الفورم مباشرة كما انه زر أو ليبل ولكنه سيظهر كما في هذه الصورة:

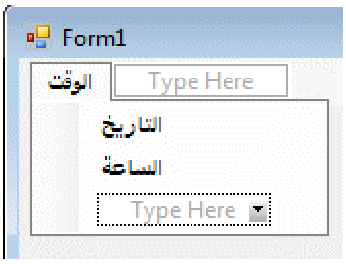


يوجد فرق هنا بين نسخة الفيجوال 6 ونسخة 2008 ففي فيجوال بيسك 6 كان مكون القوائم يظهر على الفورم وليس تحت الفورم. وكانت تظهر معظم المكونات على الفورم حتى المكونات التي لا تشاهد مثل الـــTimer وغيره من المكونات كانت تظهر على الفورم وقت تصميم المشروع. لكن نسخة 2008 تضع المكونات التي لا تظهر وقت التشغيل مثل القوائم والمؤقتات Timers في خانة خاصة أسفل الفورم تستطيع منها التعامل معهم مثل حذفهم وتغيير خصائصهم منها وليس من الفورم.

بالنسبة للقوائم ستلاحظ وجود مكان لها أعلى الفورم بعد إضافتها، تستطيع منها إضافة بنود جديدة في القوائم.



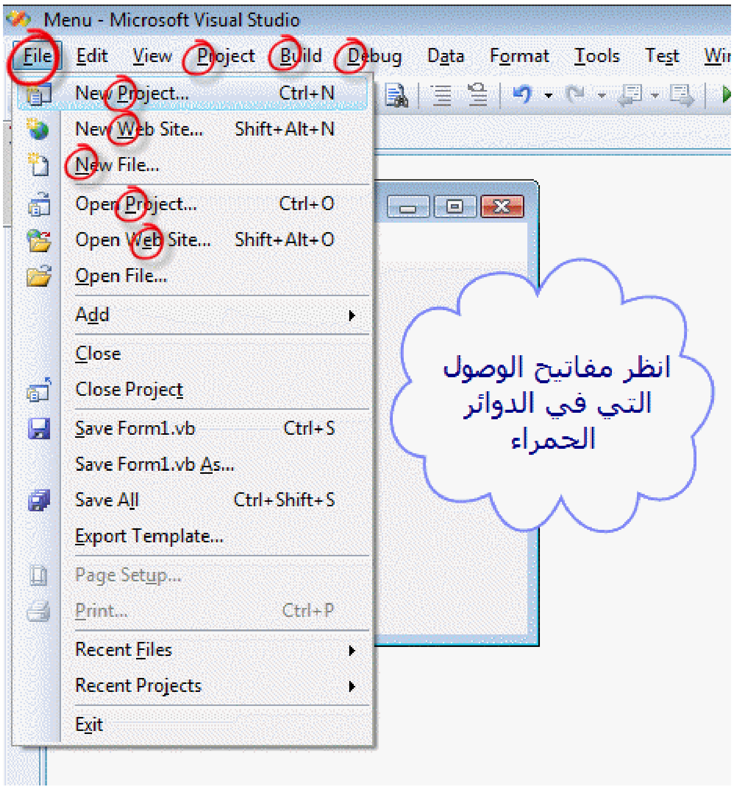
اذهب الى اعلى الفورم للقائمة عند المكان المكتوب فيه Type Here واكتب التالي في السطر الاول "الوقت" و"التاريخ" في السطر الثاني و "الساعة" في السطر الثالث ليظهر الفورم بعدها بهذا الشكل :



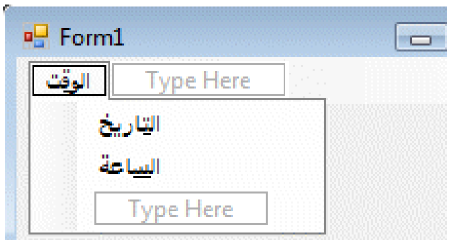
انقر فوق الفورم لتخرج من مصمم القوائم.

إضافة مفاتيح الوصول:

مفاتيح الوصول هي تلك المفاتيح أو الحروف التي تضغطها بالإضافة إلى زر الكيبورد Alt لتصل إلى زر ما من أزرار القوائم في البرنامج. فمثلا في برنامج Visual Studio 2008 تستطيع الوصول إلى قائمة File بالضغط على مفتاح الوصول F بعد الضغط على الزر Alt، بعد فتح قائمة File تستطيع الوصول إلى New Project بالضغط على الحرف P، تستطيع أن تعرف إن الحرف الفلاني هو حرف الوصول إلى الأمر المعين بمشاهدة الحرف الذي تحته الخط في ذلك الأمر، فالحرف الذي تحته الخط هو حرف الوصول إلى ذلك الأمر، انظر الصورة التالية:



لإضافة مفتاح الوصول إلى أي زر من أزرار القائمة اذهب إلى القائمة وادخل في الزر الذي تريد إضافة مفتاح وصول له وقم بإضافة العلامة & (الموجودة في الكيبورد فوق الرقم سبعه تستطيع اختيارها بواسطة Shift+7) قبل الحرف المراد اختياره مفتاح وصول. فمثلا في المثال السابق نضيف العلامة & قبل الـــ ق في الوقت وقبل الـــ ت في التاريخ وقبل الـــ س في الساعة ليصبح البرنامج بهذا الشكل:



إضافة مفاتيح الوصول للقوائم باللغة العربية قد لا يتفعل وقت تشغيل البرنامج إذا كانت الإعدادات الإقليمية للويندوز انجليزية أو إذا كان الكيبورد مقلوب باللغة الانجليزية.

لاحظ الخط الصغير الموجود تحت القاف والتاء والسين بسبب أنهم مفاتيح للوصول.

قواعد عامة لإضافة القوائم لبرنامجك

هناك قواعد عامه متعارف عليها لإضافة القوائم إلى برنامجك ومنها:

١- ابدأ كل بند من القوائم بحرف Capital في حالة القوائم باللغة الانجليزية، واحرص ان يكون كل بند عبارة عن كلمة واحدة أو اثنتين كحد أعلى.

٢- اجعل قوائمك سهلة ومفهومة واختر كلمات سهلة للتعبير عن وظائفها، لا تجعل المستخدم يحتار في وظيفة بند من بنود القائمة، فمثلاً لإغلاق البرامج استخدم الكلمة "إغلاق" ولا تستخدم كلمة غريبة مثل "روح".

٣- أضف مفاتيح للوصول لكل بند في قوائمك ويفضل أن يكون مفتاح الوصول هو الحرف الأول في البند إذا بالإمكان.

٤- قم بجعل كافة البنود المتشابهة تحت قائمة واحدة، فمثلاً البنود الخاصة بالغلق والفتح والحفظ والطباعة تحت قائمة واحده.

٥- إذا كان لديك بند من بنود القائمة يستخدم طريقة On/Off الفتح والغلق قم بإضافة زر تأشير ☑ بجانب البند فإذا كان مؤشر معناه مفتوح والعكس بالعكس.

٦- أضف ثلاث نقاط لزر القائمة الذي سيفتح لنا نوافذ حوار أو سيطلب من المستخدم معلومات إضافية في حالة ضغطه، فمثلاً زر الطباعة سيطلب من المستخدم تحديد عدد الصفحات المراد طباعتها في الورقة الواحدة وكذلك نوع الطابعة ومقاس الورق وغيرها من المعلومات الإضافية ولذلك لزم على المبرمج إضافة ثلاث نقاط ... لمثل هذه الأزرار في القائمة وقت تصميم البرنامج.

للعلم القواعد أعلاه ليست ملزمة للمبرمج لتصميم القوائم ولكن المبرمج الذي يريد تصميم البرامج المعيارية أو وفق المعايير عليه/عليها إتباع القواعد أعلاه وغيرها من معايير القوائم وانظروا إلى البرامج العالمية المشهورة ستجد إن مصمميها اهتموا بهذه المعايير.

	في الحالة العادية لا يقوم ويندوز بإظهار الخط تحت حروف الوصول أو تحت مفاتيح الوصول، وإنما يقوم بإظهار الخط بعد الضغط على الزر Alt في الكيبورد. ولكن لتفعيل هذه الخاصية (إظهار الخط دائما تحت مفاتيح الوصول) كالتالي:
Windows2000	Control Panel>Display>Effects
	لوحة التحكم ثم العرض ثم التأثيرات
Windows Xp Windows_ Server 2003	Control Panel>Display>Appearance>Effects
	لوحة التحكم ثم العرض ثم الهيئة أو الظهور ثم التأثيرات
Windows Vista	Control Panel >Appearance and Personalize> Ease Of Access Center> Make the Keyboard Easier to use> Underline Keyboard Shortcuts And Access Keys
	لوحة التحكم ثم المظهر وإضفاء الطابع الشخصي ثم مركز سهولة الوصول ثم إضفاء المزيد من السهولة على استخدام لوحة المفاتيح ثم تسطير مفاتيح

	الاختصار.

نعود لمشروعنا أعلاه MyMenu لنقوم ببعض التغييرات على القوائم. فنقوم مثلاً بتغيير ترتيب بنود القوائم فإذا أردنا جعل "الساعة" قبل "التاريخ" في القائمة نستخدم طريقة السحب والإلقاء فنقوم بسحب كلمة(بواسطة الماوس) "الساعة" ونضعها قبل التاريخ في القائمة.

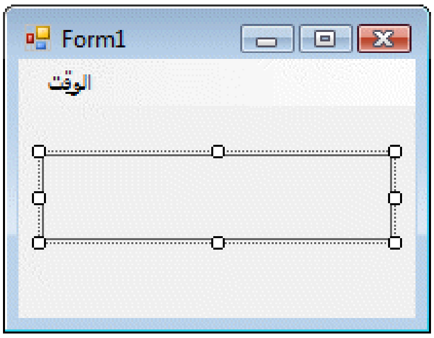
لحذف بند من بنود القائمة نحدده بواسطة الماوس ثم نضغط على Delete

مرحلة الكود

نقوم بإضافة ليبل إلى الفورم إلى وسط الفورم لعرض التاريخ والوقت عليه وقت تشغيل البرنامج فيه. ستقوم بيئة التشغيل بوضع الاسم Label1 إلى هذا الليبل الذي قمنا بإضافته، نقوم بتعديل بعض الخصائص الخاصة به كالتالي من نافذة الخصائص Properties:

الإعدادات	الخاصية
False	AutoSize
FixedSingle	BorderStyle
Tahoma, 14.25pt	Font
فارغ	Text
MiddleCenter	TextAlign

قم بتحريك الليبل إلى وسط الفورم (لأنه سيظهر الوقت والتاريخ بداخله) كما في هذه الصورة:



الآن نقوم بإضافة الكود لإظهار التاريخ وكود لإضافة الوقت في حالة اختيارهما من القائمة.

قم بعمل Double-Click على القائمة على بند "التاريخ" ثم نكتب الكود التالي:

```
Label1.Text = DateString
```

نضغط Double-Click على بند "الوقت" ثم نكتب الكود التالي:

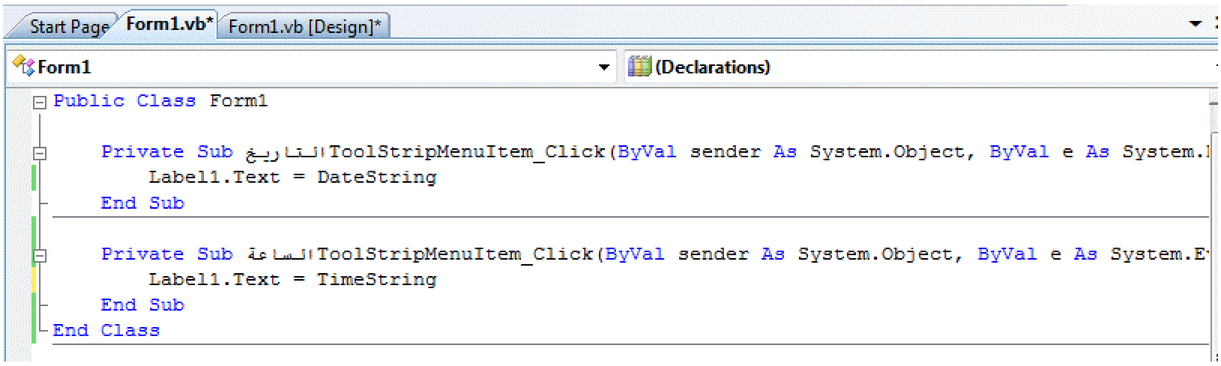
```
Label1.Text = TimeString
```

قمنا بالكود أعلاه بتغيير النص في ليبل1 إلى التاريخ في الكود الأول والى الوقت في الكود الثاني.

لاحظ أن DateString تقوم بإظهار التاريخ بحسب التاريخ الموجود ضمن نظام الويندوز وكذلك TimeString تقوم بإظهار الوقت بحسب الوقت ضمن نظام التشغيل، تستطيع تغيير وقت وتاريخ النظام من لوحة التحكم باستخدام خيارات الوقت والإعدادات الإقليمية.

الخاصية TimeString تعرض لنا الوقت الحالي وتعتبر بديل للخاصية القديمة $Time	
سوف تتعرف أكثر على الأكواد البرمجية في الفصول الخامس والسابع من هذا الكتاب	

ستكون واجهة الكود بهذا الشكل:



قم بتشغيل البرنامج ثم افتح القائمة واختر منها التاريخ ثم الساعة، ماذا تلاحظ ستلاحظ تغير النص الموجود في ليبل1 إلى التاريخ وكذلك إلى الوقت. قم بالضغط على الزر Alt ثم اضغط على الحرف س، قم بالضغط على الزر Alt ثم اضغط على الزر ت، ماذا تلاحظ ستلاحظ ظهور الوقت والتاريخ في Label1 على الفورم. هذا معناه أن المفتاحين "س" و "ت" قاما بعملية تسهيل الوصول للبندين الساعة والتاريخ في القوائم التي انشأناها.

في حالة ضغطت مفتاح Alt ثم اخترت احد الزرين "س" أو "ت" ولم يتم إظهار الساعة أو التاريخ فهذا يعني أن لغة الكيبورد ليست اللغة العربية. قم بتغيير واجهة الكيبورد إلى العربية ثم اضغط مفاتيح الوصول لأنهما باللغة العربية.

قم بإغلاق البرنامج.

ستجد نسخة من المشروع أعلاه في المرفق رقم 008، سيتم إرفاق المرفق كاملاً بعد استكمال دراسة الـ ToolStrip في الدرس التالي.

نرجع إلى الكود أعلاه الذي يقوم بإظهار التاريخ:

```
Label1.Text = DateString
```

عرفنا أن الخاصية DateString تقوم بعرض التاريخ، السؤال يقول هنا هل توجد صيغ أخرى لعرض الوقت أو التاريخ، نعم يوجد صيغ أخرى لعرض الوقت والتاريخ بطرق أخرى (مثل عرض يوم الأسبوع وكذلك الشهر أو الدقيقة) لاحظ هذا الجدول لتعرف العديد من الصيغ الهامة:

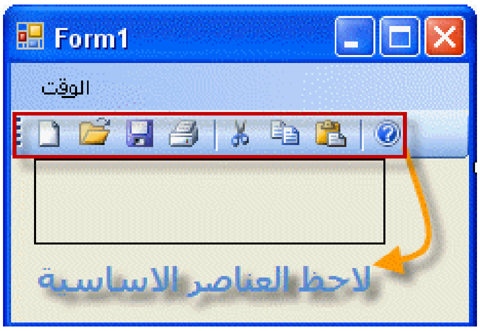
الخاصية أو الدالة	الوصف (كل البنود أدناه تعتمد على ساعة النظام)
TimeString	تظهر لنا الوقت الحالي.
DateString	تظهر لنا التاريخ الحالي.
Now	تظهر لنا الوقت والتاريخ وكذلك صباحاً أو مساءاً.
(Hour(date	رقم الساعة في اليوم بحيث نكتب الوقت بين القوسين فيقوم الكمبيوتر بتحديد رقم الساعة، طبعاً النتائج الممكنة بين 0 و 23.
(Minute(date	رقم الدقيقة في اليوم بحيث نكتب الوقت بين القوسين فيقوم الكمبيوتر بتحديد رقم الدقيقة، طبعاً النتائج الممكنة بين 0 و 59.
(Second(date	رقم الثانية في اليوم بحيث نكتب الوقت بين القوسين فيقوم الكمبيوتر بتحديد رقم الثانية، طبعاً النتائج الممكنة بين 0 و 59.
(Month(date	رقم الشهر في السنة بحيث نكتب التاريخ بين القوسين فيقوم الكمبيوتر بتحديد رقم الشهر، طبعاً النتائج الممكنة بين 1 و 12.
(Year(date	رقم السنة مثلا عام "2000" عام "2005" الخ.
(Weakday(date	يوم الأسبوع، النتائج الممكنة "السبت" أو "الأحد" الخ.

إضافة أشرطة الأدوات Toolbars باستخدام المكون ToolStrip

بإمكانك إضافة شريط للأدوات في برنامجك لتعمل بشكل لتقوم بنفس العمل التي تقوم به القوائم بحيث يختار المستخدم شريط الأدوات أو القوائم ويعتبر هذا نوع من التسهيل على المستخدم بحيث يستخدم الماوس أو الكيبورد ولديه أكثر من بديل للتعامل في واجهة المستخدم. توجد العديد من

الإضافات والميزات لأشرطة الأدوات حيث يمكنك إضافة فواصل وصناديق تأشير ☑ ومكان للنصوص وكذلك Label و ComboBox.

اختر المكون ToolStrip من قائمة الأدوات وقم بإضافته إلى الفورم، لا تهتم بمكان إضافة المكون على الفورم ستقوم بيئة التطوير بإضافة المكون إلى الفورم أوتوماتيكيا وسيظهر المكون تحت الفورم كما في المكون MenuStrip ، بعد إضافة المكون ToolStrip اذهب إلى المربع الصغير أو المثلث الصغير في أعلى يمين المكون قم بالنقر عليه ثم اختر Insert Standards Items ، ستلاحظ تكوين العناصر الأساسية في البرامج كما في هذه الشاشة:



لسنا بحاجة الآن للعناصر الأساسية، ولكن قمنا بإظهارها لمعرفة الخصائص والميزات التي تتمتع بها بيئة التطوير، لذلك نقوم بإلغاء أخر عملية Undo بالضغط على Ctrl+Z.

نقوم بإضافة اثنين Button لعناصر لشريط الأدوات، نضغط السهم الصغير في ToolStrip ونضيف الاثنين الــ Button، ثم نذهب إلى الخصائص ونسمي احدهم ClockBtn والآخر DateBtn، اذهب للخصائص التابعة لهما ثم قم باختيار الخاصية Image واختر صورة الساعة للأولى وصورة التقويم للثانية (صورة الساعة والتقويم مرفقة مع المشروع 008b). لإضافة الكود علينا أن نتذكر أننا لا نريد أن نخترع كودا جديدا وإنما نريد إضافة نفس الكود السابق للزرين المضافين بحيث يستطيع المستخدم الاختيار من القوائم أو من شريط الأدوات.

ننقر Double-click على المكون الأول في شريط الأدوات ثم نكتب في سطر الأوامر الكود التالي:

```
الساعةToolStripMenuItem_Click(sender, e)
```

ركز في الكود أعلاه من أين نقلنا الكود انه من كود القوائم الموجود في الأعلى، فبدلاً من أن نرهق أنفسنا بكتابة الكود من جديد قمنا بكتابة اسم الدالة الموجودة في الكود مسبقاً لتنقل لنا الأكواد الموجودة ضمنها مسبقاً. ننتقل إلى المكون الثاني في شريط الأدوات وبعد الضغط double-click والانتقال إلى منطقة الكود نكتب التالي:

```
التاريخToolStripMenuItem_Click(sender, e)
```

نفس الطريقة التي كتبنا بها الكود بالأعلى.

نقوم بتشغيل البرنامج ونجرب استخدام الأزرار الموجودة في شريط الأدوات، سنلاحظ ظهور التاريخ وكذلك الوقت في المكان المحدد في الفورم.

الآن سنذهب بعيداً بعض الشيء عن التاريخ والوقت، تعالوا لنجرب تغيير لون النص داخل الليبل (طبعاً النص سيكون تاريخ أو وقت). لتغيير اللون داخل الليبل ومن أجل الحصول على العديد من الخيارات لابد من استخدام ميزة جديدة من ميزات الفيجوال 2008 ألا وهي نوافذ الحوار Dialog Box Controls .

استخدام نوافذ الحوار Using Dialog Box Controls

يحتوى الفيجوال 2008 على ثمان نوافذ Dialog Box Controls حوار جاهزة للاستخدام. للعلم فهي معده لك مسبقاً لذلك فلا تحتاج إلى إعدادها من جديد وتستخدم للمهمات المشهورة والمتكررة مثل نافذة فتح ملف أو غلقه أو طباعته. لن تحتاج إلا أن تقوم بإدخال كود للأحداث المتوفرة ضمن المكون (نافذة الحوار)، أما بالنسبة للتصميم فقد تم تصميمها مسبقا بحسب المعايير الموجودة مع

نظام التشغيل الويندوز. للعلم هذه المكونات الثمانية تشابه المكون CommonDialog الذي كان متوفر مع فيجوال بيسك 6 ، إليك النوافذ الحوارية الثمانية المتوفرة ضمن بيئة التطوير 2008:

اسم المكون	الهدف منه
OpenFileDialog	للحصول على ملف أو امتداد معين من الملفات من قرص معين من مجلد معين في الكمبيوتر للملفات الموجودة مسبقا.
SaveFileDialog	تحدد اسم القرص واسم المجلد وكذلك اسم الملف للملف الجديد.
FontDialog	تسمح للمستخدم من اختيار نوع الخط وطريقة عرضه.
ColorDialog	تسمح للمستخدم اختيار لون محدد من مجموعه ألوان.
FolderBrowserDialog	تسمح للمستخدم من التنقل بين المجلدات واختيار مجلد معين.
PrintDialog	تسمح للمستخدم بتغيير خيارات الطباعة.
PrintPreviewDialog	تسمح للمستخدم بمعاينة المواد التي يريد طباعتها قبل الطباعة كما يفعل برنامج الورد.
PageSetupDialog	تسمح للمستخدم بتغيير خيارات الصفحة بتغيير الحدود للصفحة وكذلك حجم الورق وغيرها من الإعدادات.

في المثال التالي سنتعرف على كيفية استخدام اثنين من المكونات أعلاه وهما OpenFileDialog وكذلك ColorDialog المكون الأول سيسمح لك بتحميل صورة إلى الفورم والمكون التالي سيسمح لك بتعيير نوع خط صندوق نص معين.

أنشئ مشروع جديد وقم بإضافة المكونات التالية للفورم : TextBox و عدد اثنين Button وكذلك OpenFileDialog و ColorDialog ثم أضف PictureBox وقم بترتيبهم على الفورم وتنظيمهم بشكل مناسب.

قم بضبط الخاصية SizeMode في PictureBox على StretchImage.

و اضبط بعض الخصائص كالتالي:

الكائن	الخاصية	التعديل اللازم
Button1	Text	أستعرض الصور
OpenFileDialog1	FileName	فاضي (بدون نص)
Button2	Text	اختر لون الخط

مرحلة الكود

قم بإضافة هذا الكود للإجراء Click التابع للزر Button1 (لاحظ: تم شرح الكود بين السطور):

```
 ' نضع مرشح فلتر لإختيار نوعية محددة من الملفات فقط
      OpenFileDialog1.Filter = "Bitmaps (*.bmp)|*.bmp"
      ' إذا تم إختيار صورة معينة بواسطة نافذة الحوار التي ظهرت فقم بالتالي
      If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
          ' ضع صورة في صندوق الصورة ولتكن هذه الصورة هي الصورة المختارة من نافذة الحوار
          PictureBox1.Image = System.Drawing.Image.FromFile _
          (OpenFileDialog1.FileName)
      End If
```

لاحظ أننا عملنا مرشح لامتداد الملفات bmp فقط وعلية فلا يمكن اختيار ملف من غير هذه النوعية من الملفات. ثم حولنا الصورة المختارة إلى صندوق الصورة الموجود معنا.

صندوق الصور المتوفر ضمن بيئة التطوير يظهر الامتداد التالية من الصور: bmp, emf, wmf, ico, jpg, jpeg, png, gif،

ولإضافة امتداد أخر للفلتر أو المرشح فنقوم بإضافة الامتداد بعد الحرف | مثل هذا الكود:

```
OpenFileDialog1.Filter = "Bitmaps (*.bmp)|*.bmp|Metafiles (*.wmf)|*.wmf"
```

فالمرشح في الكود أعلاه سيسمح للامتدادين bmp و wmf بالظهر فقط.

السطر الثاني في الكود يظهر نافذة الحوار التي تسمح لنا باختيار الصورة المراده، لاحظ أننا استخدمنا ShowDialog كبديل عن Show التي كنا نستخدمها في فيجوال 6 لكن ShowDialog تستخدم لإظهار أي فورم. ShowDialog ترجع لنا نتيجة وهي اختيار صورة محدده أو ملف معين أو عدم اختيار وتسمى هذه النتيجة DialogResult. فقد تكون هذه النتيجة اختيار صورة أو ملف معين DialogResult.OK وعليه فينفذ الأمر الذي أعطيناه وقد تكون النتيجة DialogResult.None فلا يقوم بتنفيذ الأمر الذي كتبناه له لأننا كتبنا له الأمر بصيغة If بمعنى إذا الشرطية. ستعرف الكثير عن الجملة الشرطية If في الفصل السادس من هذا الكتاب.

ألان سننتقل لكتابة كود لنافذة اختيار الألوان: Double-click على الزر Button2 والذي هو لاختيار لون الخط ثم نكتب الكود التالي بين Private Sub و End Sub :

```
' إظهار مربع الالوان
        ColorDialog1.ShowDialog()
        ' لتغيير لون النص الموجود في مربع النص الى اللون المختار
        TextBox1.ForeColor = ColorDialog1.Color
```

اضغط على F5 لاستعراض البرنامج. التحكم باللون المختار بتحديد خيارات مربع الألوان

تستطيع تحديد الألوان المعروضة في مربع الألوان بواسطة تحديدها من نافذة الخصائص لمربع الألوان ColorDialog أو بواسطة الكود قبل السطر الذي فيه ShowDialog في الجدول التالي تستعرض بعض الخصائص الخاصة بمربع الألوان ColorDialog . للعلم كل خاصية من الخواص التالية تستطيع تفعيلها بواسطة اختيار True أو عدم تفعيلها بواسطة اختيار False

AllowFullOpen	تضبط على True لتسمح بالانتقال إلى وضع الألوان كاملة للاختيار منها.
AnyColor	تضبط على True للسماح للمستخدم باختيار أي لون من مربع الألوان
FullOpen	تضبط على True إذا أردت إظهار كل الألوان من البداية بما فيها الألوان المخفية تحت Custom Colors.
ShowHelp	تضبط على True إذا أردت إظهار زر التعليمات Help في مربع الألوان.
SolidColorOnly	تضبط على True إذا أردت اختيار الألوان الــ Solid فقط أما الألوان التي تنشأ من تداخل بعض بكسلات الألوان الأخرى سيتم إهمالها.

ألان تستطيع تشغيل البرنامج وللملاحظة يوجد نسخة منه في المرفقات المرفق رقم 009، للعلم تستطيع إضافة بعض التعديلات على البرنامج المرفق مثلاً قم بتغيير طريقة عرض الألوان بالاستفادة من الجدول أعلاه للخيارات الممكنة لعرض الألوان وكذلك تستطيع إضافة امتداد جديد لملفات الصور التي ممكن أن يعرضها مستعرض ملفات الصور بالطريقة التي حددناها سابقاً.

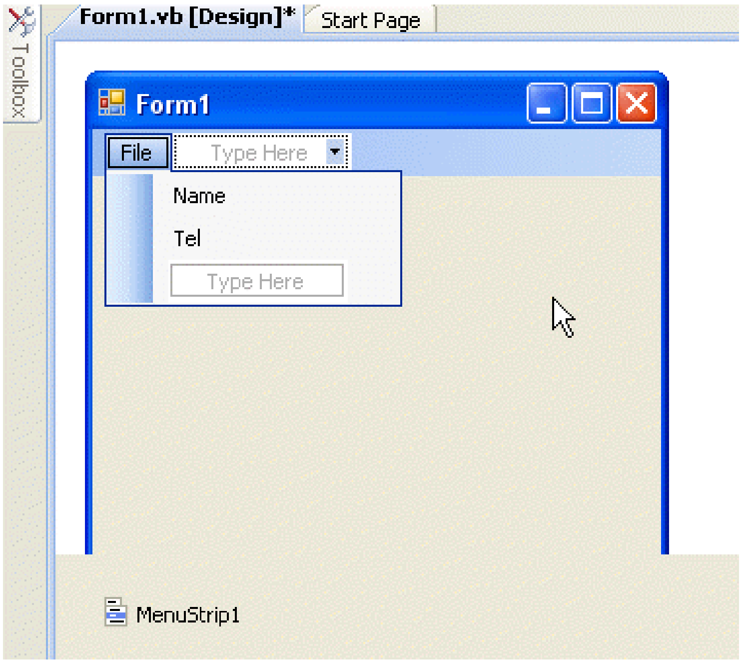
إضافة نوافذ حوار غير متوفرة ضمن بيئة التطوير

تستطيع إضافة صناديق حوار أو نوافذ حوار غير تلك المتوفرة ضمن بيئة التطوير بواسطة إضافة Forms إلى برنامجك وإضافة بعض الأكواد لمعرفة المدخلات والمخرجات Inputs and Outputs ستتعرف على المزيد في هذا الجانب في الفصل الرابع عشر وكذلك الفصل الخامس من هذا الكتاب.

فائدة: إضافة اختصارات إلى برنامجك

في برامج الأوفس (الورد والأكسل) وغيرها من البرامج ستلاحظ وجود اختصارات للعديد من المهام ، بحيث تستطيع الوصول إلى تلك المهمة بدون الذهاب الماوس إلى القوائم ثم إلى المهمة

المعنية. فمثلاً إذا أردت نسخ نص معين في برنامج من برامج الأوفس فإننا نقوم بتظليل ذلك النص ثم نقوم بالذهاب إلى القوائم Edit ثم نختار Copy. ولاختصار هذه المهمة الطويلة نقم بالضغط على الاختصار Ctrl+C. ونريد الآن أن نتعرف كيف يتم تعيين هذه النوعية من الاختصارات في برنامجنا. نقوم بإنشاء مشروع جديد ثم نضيف المكون MenuStrip له ونضيف قائمة File في المكون ثم نضيف لها قائمتين فرعيتين وهما Name و Tel ليصبح مشروعنا كما في الصورة:



نضغط Double-Click على Name ثم نكتب الكود التالي لإظهار مربع نص للمستخدم كالتالي:

```
MsgBox("Marwan")
```

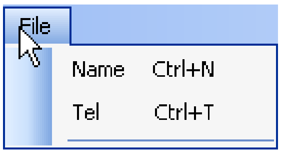
نضغط Double-Click على Tel ونضيف الكود التالي:

```
MsgBox("00967*******")
```

نجرب البرنامج بالضغط على F5 ثم نختار Name ماذا نلاحظ ثم نختار Tel ماذا سنلاحظ.

بعدها نغلق البرنامج ثم نذهب إلى واجهة التصميم. نختار القائمة الفرعية Name ثم نذهب إلى نافذة الخصائص التابعة للمكون نذهب إلى الخاصية ShortCutKeys نختار Ctrl ثم الحرف N

كذلك نختار المكون Tel ثم نذهب إلى الخصائص الخاصية ShortCutKeys نختار Ctrl ثم الحرف T. بعدها نقوم بتشغيل البرنامج بالضغط على F5 ثم نضغط على Ctrl+T سنلاحظ أن البرنامج يقوم بعرض صندوق الحوار الذي كان يظهره عندما نختار القائمة الفرعية Name بواسطة الماوس وذلك لأننا حددنا المفتاحين Ctrl والمفتاح T كاختصار لتنفيذ الأمر المتوفر ضمن القائمة Name . ملحوظة توجد نسخة من البرنامج في المرفق 010 .

في الحالة العادية يقوم فيجوال بيسك 2008 بإظهار مفاتيح الاختصارات بجانب المكون (في حالة توفرها) مثل هذه الصورة 

ولكن إذا لم نرد إظهار هذه المفاتيح بجانب المكون نقوم بضبط الخاصية ShowShortcutKeys على False في هذه الحالة ستبقى مفاتيح الاختصارات مفعلة إذا ضغط عليها المستخدم ولكن لن تكون ظاهرة أمامه. إذا أردت إظهار نص أخر بدل حروف الاختصار تستطيع ذلك بكتابة النص المراد ظهوره في الخاصية ShortcutKeyDisplayString التابعة للمكون.

خلاصة الفصل الرابع:

من اجل أن	قم بالتالي
لإنشاء بند في القوائم	أضف المكون MenuStrip إلى الفورم ثم اذهب إلى Type Here في أعلى الفورم وأضف القائمة التي تريد وأضف لها فروع بالشكل المناسب.
إضافة مفتاح وصول للبند	Double-Click على البند المراد وأضف الحرف & قبل الحرف المراد جعله مفتاح للوصول.
إضافة اختصار للبند	حدد البند ثم اذهب إلى الخصائص واضبط الخاصية ShortcutKeys على الاختصار الذي تريد.

تغيير ترتيب عناصر القوائم	بواسطة الماوس بطريقة السحب والإلقاء.
إضافة شريط أدوات	أضف المكون ToolStrip إلى الفورم وقم بعمل Right-Click على الأزرار لتنظمهم وتضيف أزرار أخرى ثم Double-Click على أي بند لكتابة الكود الخاص به.
استخدام نافذة حوار	أضف واحدة من الثمانية المكونات التابعة لنوافذ الحوار (وتوجد تحت البند Dialogs and Printing Toolbars مع بقية المكونات)إلى الفورم وقم بتغيير خصائصها من نافذة الخصائص.
استخدام نافذة حوار لفتح نوع من الملفات	أضف المكون OpenFileDialog إلى الفورم، لإظهار نافذة الحوار لابد من استخدام الطريقة ShowDialog Method. الخاصية FileName تحتوي على اسم الملف الذي تم اختياره.
استخدام نافذة حوار لفتح صندوق الألوان	أضف المكون ColorDialog إلى الفورم، لإظهار صندوق الألوان نستخدم الطريقة ShowDialog Method. الخاصية Color تحتوي على اللون الذي أختاره المستخدم.

الجزء الثاني: أساسيات البرمجة

من الفصل الخامس إلى الفصل الثالث عشر

في الجزء الأول من هذا الكتاب "**إبدأ مع الفيجوال بيسك 2008**" تعلمنا كيف نبني برامج وكيف ننظم واجهة البرنامج للمستخدمين وكذلك كيف نعدل على البرنامج وكيف ننشره. تعلمنا كذلك كيف نتعامل مع بيئة التطوير. سيكون عنوان الفصول التسعة القادمة "**أساسيات البرمجة**" سوف تعرف

الكثير عن مرحلة الكود في فيجوال بيسك وكيف يتعامل المعالج مع الأوامر البرمجية. سوف تتعرف على كيفية استخدام جمل الشرط والمؤقتات والمصفوفات وجمل الدوران Loops والتعامل مع الملفات النصية. ستتعلم أيضا كيف تتعامل مع الـــ Debug (أو تجميع وتنقيح وتشغيل) البرنامج وكذلك كيف تتعامل مع أخطاء التشغيل بعد الفصل الثاني ستكون ذو قابلية اكبر للتعامل مع المواضيع المتقدمة في البرمجة لذلك ستتعرف بعد الفصل الثاني إن شاء الله على تنظيم المكونات في واجهة المستخدم، برمجة قواعد البيانات وكذلك على برمجة مواقع الانترنت.

الفصل الخامس: المتغيرات والدوال وبيئة الدوت نت.

الجمل البرمجية في فيجوال بيسك:

كما تعلمنا في الفصل الثاني "كتابة برنامجك الأول" سطر الأوامر في فيجوال بيسك يسمى الجملة البرمجية Program Statement فالجملة البرمجية: هي عبارة عن ترابط منسق من الكلمات والخصائص والمكونات والمتغيرات والأرقام والمعاملات الخاصة والقيم الأخرى التي ترتب بشكل منطقي لتصنع أمر برمجي معين مفهوم لدى المترجم للغة الآلة الـــ Compiler. قد تكون الجملة البرمجة عبارة عن كلمة واحدة مثل:

End

والتي تقوم بإغلاق البرنامج. أو قد تكون الجملة البرمجية عبارة عن مجموعة من المكونات المذكورة أعلاه مثل الجملة التالية:

Label1.text = TimeString

حيث أسندنا الخاصية Text التابعة للمكون Label1 إلى الطريقة TimeString التي تحتوي على الوقت الحالي، وبذلك سيتم إظهار الوقت الحالي في المكون Label1. القواعد أو القوانين التي لابد أن نتبعها من أجل كتابة جمل برمجية صحيحة تسمى Statement Syntax أو بناء الجملة. هذه القواعد هي نفسها مع بعض التغيير القواعد القديمة الموجودة في النسخ القديمة من

فيجوال بيسك 2008 وكذلك يتشارك فيجوال بيسك مع اللغات البرمجية الأخرى مع بعض الاختلافات اللازمة. لكي نتعلم بناء جملة برمجية صحيحة لابد من الاطلاع على هذه القوانين أو أساسيات بناء الجمل البرمجية وكيفية معالجة البيانات ضمن البرنامج لمكتب جملة برمجية صحيحة. وبصراحة لغة الفيجوال بيسك هي لغة برمجية سهلة تسهل على المبرمجين العديد من الصعاب لذلك فبناء برنامج بها سهل جداً ويكون قريب من اللغة العامية في بعض الأحيان، كل هذا من اجل التسهيل على المبرمجين وجعلهم يفرغون عقولهم للأفكار الجديدة والتطويرية فبدلاً من كتابة صفحتين من الكود لإنشاء فورم، تتم العملية فقط بواسطة السحب الإلقاء بواسطة الماوس، وهناك العديد من الوسائل التي تبسط لنا البرمجة بالفيجوال بيسك. في نفس الوقت بيئة التطوير تساعدك في تحديد الأخطاء وتقديم الحلول الممكنة أو المقترحات الممكنة للمبرمج. في هذا الفصل وفي الفصول القادمة إن شاء الله ستتعرف على الكائنات والدوال والكلمات والطرق والخصائص الموجودة مسبقاً في بيئة الدوت نت وسنتعلم كيف نستفيد منها لتطوير وتصميم برامج عملاقة. أول موضوع لنا في هذه السلسلة لابد من معرفة المتغيرات وأنواع البيانات لأنهما خطوة هامة جداً لمواصلة التعلم في هذا الكتاب.

استخدام المتغيرات لحفظ البيانات:

المتغير: هو مكان مؤقت لحفظ البيانات في برنامجك. تستطيع استخدام متغير واحد أو أكثر في برنامجك وقد تكون هذه المتغيرات كلمات أو أرقام أو تواريخ أو خصائص أو قيم أخرى مثل قيم النظام الثنائي Binary (صفر وواحد). باستخدام المتغيرات تستطيع تسمية كل نوع من أنواع البيانات باسم سهل التذكر ذو معنى مفيد ليساعد على تسهيل عملية البرمجة وتقوم المتغيرات بحفظ البيانات التي يدخلها المستخدم أو يتم جلبها من النظام أو من الشبكة أو غيرها من المصادر وقت عمل البرنامج Run-Time وقد تكون المتغيرات عبارة عن بيانات تمت معالجتها ببرنامجنا وقت عمل البرنامج نستطيع أن نستعرض البيانات المخزونة في المتغيرات على الفورم أو خزنها في قاعدة البيانات (خزنها بشكل دائم) لان المتغيرات تخزنها بشكل مؤقت فقط لحين إغلاق البرنامج

أو للوقت الذي نحدده نحن. استخدام المتغيرات في بيئة التطوير يلزمنا بتخطيط لمعرفة ما هي المتغيرات التي نحتاجها لان حجز المتغيرات في برنامجنا مثل حجز كرسي في قاعة المحاضرة فلا نقوم بحجز الكرسي إلا إذا كنا محتاجين له سوف نتعرف على كيفية حجز مكان للمتغيرات في الفصل قريباً.

تعريف المتغيرات بواسطة الكلمة Dim:

منذ نسخة فيجوال بيسك 2003 كان لابد على المبرمجين من تعريف المتغيرات قبل استخدامها في البرنامج، طبعاً هذا يعتبر تغيير في إستراتيجية فيجوال بيسك في فيجوال 6 والنسخ الأقدم حيث كان باستطاعتنا (وتحت شروط معينة) استخدام المتغيرات التي عرفناها بطريقة ضمنية – أي بدون استخدام الكلمة Dim. الأسلوب القديم كان سهل ومرن للتعامل لكن كان فيه العديد من المخاطر التي واجهها المبرمجين حيث كثرت الأخطاء في طباعة اسم المتغيرات وكذلك الارتباك الذي يحصل بعدم تعريف المتغيرات مما يجعل البرامج تحتوي على ثغرات وأخطاء برمجية لم تكتشف في وقت البرمجة وإنما سببت العديد من المتاعب للمستخدمين.

هناك إمكانية في فيجوال بيسك لتعريف المتغيرات ضمنيا كما كان في النسخ القديمة لكننا لن نوضح هذه الخاصية حتى نتأكد من أننا وصلنا إلى مرحلة متقدمة حتى نعرف جدوى وعدم جدوى هذه الطريقة من طرق تعريف المتغيرات.

لنعرف متغير في فيجوال بيسك لا بد من استخدام الكلمة Dim (اختصار لـــ Dimension) ومعناها برمجيا عرف أو اعتبر أن هذه الكلمة تأمر الكمبيوتر بحجز مكان في الذاكرة للمتغير وتسمح للكمبيوتر بمعرفة نوع البيانات التي سيتعامل معها. نستطيع تعريف المتغيرات في منطقة من الكود بشرط واحد وهو تعريف المتغير قبل استخدامه، معظم المبرمجين يفضلون تعريف المتغيرات بشكل جماعي في بداية كل عملية برمجية.

مثال: هذه الجملة البرمجية تقوم بتعريف متغير اسمه LastName ونوعه: متغير نصي،

```
Dim LastName as String
```

لاحظ في التعريف أعلاه الكلمة الأولى Dim وهي لازمة لعملية التعريف بعدها مباشرة LastName اسم المتغير بعدها كلمة As ومعناها كـــ بعدها مباشرة نوع المتغير String ومعناه متغير نصي (المتغيرات النصية هي كل المتغيرات التي تحتوي على كلمات مثل الأسماء، الأماكن، الرموز الخاصة، سطور من أي قصيدة أو مقالة، محتويات ملف، أو أي بيانات نصية) بالطبع هناك أنواع أخرى من المتغيرات مثل التواريخ والأعداد وغيرها من الأنواع. نحدد نوعية المتغير لنحجز المكان المناسب له في الذاكرة المؤقتة الرام، طبعا في البرامج الكبيرة عملية خزن البيانات في الرام قد ينهك موارد النظام ويصل إلى مرحلة العجز عن معالجة البيانات لأنها اكبر من سعة الذاكرة المؤقتة الرام (للعلم كل نوع من أنواع البيانات (الأعداد، النصوص، التواريخ) يحجز مساحة معينة من الرام لذلك لابد من التنبه لهذا وقت البرمجة لكي لا نثقل على الذاكرة).

> في النسخ القديمة من الفيجوال بيسك كنا نستطيع تعريف المتغيرات بدون تحديد نوعها بحيث تعتبر أنها متغيرات من النوع العام حيث تقبل أي نوع من أنواع البيانات وتسمى Variant أي متغيرات بديلة، نسخة 2008 لا تدعم المتغيرات البديلة بشكل مباشر، بالرغم من سهولتها وإمكانية إسناد أي نوع من أنواع البيانات لها لكنها تسبب التأخر في برنامجك حيث يكون البرنامج بطئ وقت التشغيل. تستطيع استخدام هذه المتغيرات البديلة في نسخة 2008 بإسناد البيانات إلى النوع Object ليتم معاملتها وكأنها بيانات ثنائية Binary (أو صفر وواحد) حيث يتم استخدام هذا النوع من البيانات لكل أنواع البيانات بشكل عام.

بعد تعريف المتغير تكون لك الحرية بإسناد البيانات إليه باستخدام الإشارة = بعد المتغير مثل الجملة التالية التي تخزن الاسم "القحطاني" في المتغير LastName الذي قمنا بتعريفه سابقاً.

```
LastName = “القحطاني”
```

وبما إن المتغير متغير نصي نستطيع إسناد أي كلمة أو جملة أو سطر أو نصف صفحة إلى المتغير كما في المثال التالي:

LastName = "لعمرك ما ضاقت بلاد بأهلها   ولكن أحلام الرجال تضيق"

أو قد نسند عنوان مثلاً:

LastName = "صندوق بريد 15264 الرمز البريدي 22635"

وبالرغم من وجود الرقم في النص أعلاه إلا أن البرنامج سيعتبره نص لأننا عرفنا المتغير LastName بأنه نص ولكننا يمكننا التحويل بين أنواع المتغيرات باستخدام العديد من الدوال المتوفرة ضمن بيئة التطوير.

لنأخذ مثال جديد:

```
Dim LastName as String
LastName = "القحطاني"
Label1.Text = LastName
```

ففي المثال أعلاه قمنا بإسناد كلمة القحطاني إلى المتغير النصي LastName ثم جعلنا النص في Label1 يساوي المتغير LastName وعليه فالاسم "القحطاني" سوف يظهر في Label1.

التعريف الضمني للمتغيرات:

إذا أردت تعريف المتغيرات بشكل ضمني كما كان قديماً، فلابد عليك من ضبط بعد الإعدادات الخاصة بالكود، يمكننا ذلك بواسطة تعديل طريقة تعامل المترجم الـــ Compiler مع البرنامج بالطريقة التي تم توضيحها في بداية هذا الكتاب في موضوع: "**إعدادات هامة يجب التعرف عليها**"

أو نكتب الجملة البرمجية التالية في أعلى منطقة الكود:

```
Option Explicit Off
```

طبعاً هذه ليست الطريقة المثلى للتعامل مع المتغيرات ولكن قد تكون ضرورية في بعض الحالات.

وكذلك يمكنك إضافة الكود التالي لمنطقة الكود (وتعتبر ميزة جديدة في نسخة 2008):

```
Option Infer On
```

وتقوم بتقريب نوع المتغير إلى اقرب نوعية ممكنه، وذلك بمحالة استقرا مخرجات المتغير وتحديد نوعه، فمثلاً هذه الجملة البرمجية:

```
Dim attendance = 100
```

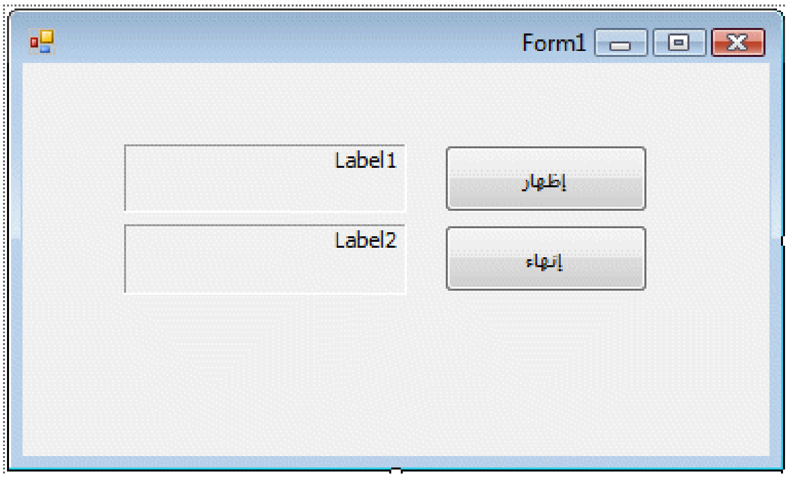
سيتم اعتبار المتغير attendance كعدد صحيح Integer لان الرقم 100 عدد صحيح، بعبارة أخرى: ضبط Option Infer على On مثل كتابة الجملة السابقة:

```
Dim attendance As Integer = 100
```

طيب، لنفرض أننا عرفنا المتغير بدون تحديد نوعه، وضبطنا Option Infer على Off ماذا سيحدث، ستقوم بيئة التطوير باعتبار المتغير من النوع Object (متغير عام)، لكن تعريف كل المتغيرات على أنها متغيرات عامة يستهلك ذاكرة الجهاز ويسبب لنا العديد من المتاعب لاحقاً.

استخدام المتغير في برنامجك:

افتح المشروع الموجود ضمن المرفق رقم 011، إذا لم يظهر اذهب إلى Solution Explorer ثم Right-Click على Form1.vb واختر View Designer. سيكون البرنامج بالشكل التالي:



ستلاحظ في الشكل أعلاه ظهور Label وكأنه صندوق، تم ذلك بتغيير الخاصية BorderStyle التابعة له إلى Fixed3D.

قم بالضغط Double-Click على الزر إظهار، سينقلك محرر الكود إلى الحدث Click التابع للزر Button1 واكتب الكود التالي:

```
Dim LastName As String

LastName = "علي"
Label1.Text = LastName

LastName = "بن علي"
Label2.Text = LastName
```

الكود أعلاه مقسم إلى ثلاث مجموعات السطر الأول لتعريف المتغير على أساس انه من النوع String ، السطرين الثاني لإسناد النص إلى المتغير السطر الثالث لإظهار النص في Label1. السطرين الأخيرين نفس الثاني والثالث مع تغيير مكان إظهار النص. ملاحظة هنا بعد تعريف المتغير وقبل استخدامه نلاحظ تكون خط أخضر متعرج تحت اسم المتغير يظهر ذلك الخط ليبين لنا وجود متغير لم يتم استخدامه، إذا قمت باستخدام المتغير في بيئة التطوير ولكن مازال الخط الأخضر المتعرج موجود تحت المتغير فهذا يعني انك أخطأت في طباعة المتغير في مكان ما في

الكود. ملاحظة أخرى، عند تعريف المتغيرات ذات القيم النصية فيجب كتابة القيمة النصية بين علامتي اقتباس "" إلا في حالة واحدة فقط وهي كون القيمة النصية أرقام فلا نحتاج لعلامتي الاقتباس. قم بعمل Double-Click على الزر إنهاء واكتب الكود التالي:

```
End
```

الكود أعلاه يقوم بإغلاق التطبيق وقت التشغيل. قم بحفظ التغييرات وتنفيذ البرنامج ثم اختر إظهار، سيقوم البرنامج بإظهار القيم النصية التي حددناها له في مرحلة الكود. انظر الصورة:



عند تحديد أسماء للمتغيرات لابد من التنبه لهذه النقاط من اجل سهوله التعامل مع المتغيرات في التطبيقات العملاقة والتي تحوي العديد من المتغيرات:

١. يجب أن يبدأ اسم المتغير بحرف أو _ علامة سطريه، لان المتغيرات في فيجوال بيسك تتكون من حروف وعلامات سطريه فقط _ وأرقام.

٢. في فيجوال بيسك 6 كان عدد الحروف في المتغير الواحد لا يسمح بزيادتها عن 256 حرف لكن في نسخة 2008 زالت هذه القيود وبالرغم من ذلك فا الأفضل أن تكون المتغيرات قصيرة ومفهومه ويفضل أن لا يتجاوز عدد الأحرف فيها عن 33 حرف.

٣. لابد أن تكون أسماء المتغيرات معبرة عن استخداماتها وان لزم ذلك دمج كلمتين مثل المثال السابق حين عرفنا المتغير بالكلمة LastName.

٤. استخدم خليطا من الحروف والأرقام والعلامات السطريه _ في تعريفك للمتغيرات ويفضل جعل الحرف الأول كبتل Capital والبقية سموول small (للعلم بعض المبرمجين يفضل جعل الحرف الأول من المتغير سموول وإتباع طريقة سنامه الجمل في تعريف المتغيرات بحيث يكون الحرف الأول سموول والحروف الأولى من الكلمات في داخل المتغير كبتل مثل المتغير dateOfBirth).

٥. لا تستخدم الكلمات المحجوزة في فيجوال بيسك (مثل الكلمات Dim، If) أو أسماء الخصائص أو أسماء الكائنات، وإلا سيقابلك خطأ ما وقت تشغيل البرنامج.

٦. لجودة أكثر في برنامجك يفضل بداية اسم كل متغير بثلاثة حروف تعبر عن نوعية بيانات المتغير فمثلاً يمكنك تعريف متغير بالاسم strName أو intLength، طبعاً هذه الطريقة ليست لدرجة كبيرة مهمة ولكنك قد تجدها بشكل كبير في أوقات معينة، وللعلم أول من استخدم هذه الطريقة هو بواسطة مبرمج في مايكروسوفت اسمه شارلز سيموني وتسمى الطريقة المجرية للتسمية.

٧. تستطيع تسمية المتغير باستخدام حروف اللغة العربية ولكن لعدم معرفة مضاعفات مثل هذه العملية في المستقبل فيفضل أن يكون المتغير باللغة الانجليزية فقط.

استخدام المتغيرات لحفظ المدخلات:

معظم الأمثلة التي تعاملنا معها سابقاً كانت حفظ المدخلات في صندوق نص TextBox وحفظنا البيانات فيها في خاصية Text، لكن في بعض الأحيان نريد أن نحفظ المدخلات في مكان أخر وليس في خاصية فلذلك نستخدم المتغيرات، أحد الطرق لجلب المدخلات من المستخدم هو صندوق المدخلات InputBox سنقوم بإظهار صندوق المدخلات للمستخدم ثم حفظ النص الذي يدخله المستخدم في متغير في المثال التالي.

أنشئ مشروعا جديدا وليكن اسمه InputBox

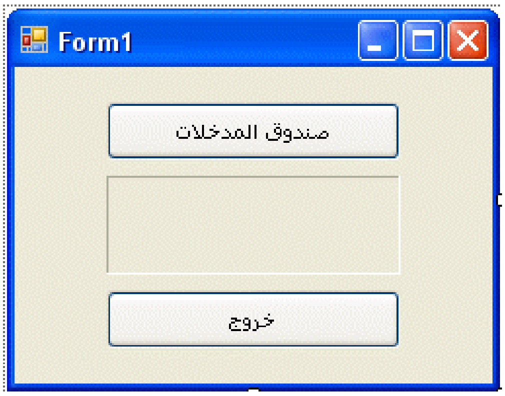
أضف اثنين أزرار للفورم واجعل خاصية الـــ Text لأحدهما "صندوق المدخلات" وللأخر "خروج"، ثم أضف ليبل للفورم ولتكن خصائصه كالتالي:

AutoSize = False

BorderStyle = Fixed3D

= Text

سيكون برنامجك كما في الشكل التالي:



قم بالضغط Double-Click على الزر "صندوق المدخلات" ثم اكتب الكود التالي:

```
Dim Prompt, FullName As String
Prompt = "قم بإدخال اسمك الكريم هنا"
FullName = InputBox(Prompt)
Label1.Text = FullName
```

في الكود السابق قمنا بتعريف اثنين من المتغيرات باستخدام الكلمة Dim وهما Prompt و FullName وجعل نوعه قيمهما قيم نصية. تستطيع تعريف أكثر من مائة متغير في سطر واحد بحيث تفصل بين كل متغير وآخر بالفاصلة ، في فيجوال بيسك نسخة 6 إذا كتبنا الكود أعلاه فلن يتقبلها الكمبيوتر بالشكل الصحيح ولكن سيفهم أن المتغير FullName هو متغير نصي أما المتغير Prompt فهو متغير من النوع العام Variant لأنه ليس متبوعا بالكلمة String. لكن

هذه المشكلة المنطقية قد تم حلها من نسخة 2002 وصاعداً. السطر الثاني من الكود أعلاه تم إسناد قيمة نصية للمتغير Prompt وتم اعتماد هذا المتغير كـــ ناقل وسيط للقيمة النصية Argument إلى الدالة InputBox وتسمى المتغيرات التي تقوم بمثل هذا العمل Arguments. في السطر الثالث من الكود يقوم البرنامج بحفظ النص المدخل من المستخدم في المتغير FullName أما في السطر الأخير فيقوم البرنامج بإظهار النص المحفوظ في المتغير FullName في الليبل Label1 علينا الآن أن نعرف أن الدالة InputBox هي دالة من دوال الفيجوال بيسك ولديها العديد من القدرات وبإمكاننا استغلال العديد من الخصائص لهذه الدالة، للمزيد من المعلومات حول هذه الدالة راجع التعليمات الخاصة ببيئة التطوير Documentation.

قم بالضغط Double-Click على الزر "خروج" ثم اكتب الكود التالي للخروج من البرنامج:

```
End
```

أحفظ التغييرات التي قمت بها ثم اضغط على F5 لتشغيل البرنامج، ثم اضغط على زر صندوق المدخلات واكتب اسمك في صندوق المدخلات الذي سيظهر ثم اضغط Enter ستلاحظ أن البرنامج أضاف اسمك أو النص الذي قمت بإدخاله في Label1. اضغط على الزر "خروج" للخروج من البرنامج.

ما هي الدالة:

الدالة هي عبارة عن مجموعة من الجمل البرمجية التي تقوم بعمل برمجي محدد و منظم وله معنى (مثلاً طلب معلومات معينة من المستخدم أو حساب محيط الدائرة بمعلومية قطرها) وترجع النتيجة إلى البرنامج، النتيجة المرجعة بواسطة الدالة يمكن إسنادها إلى متغير معين (كما في المثال أعلاه) أو يمكن إسنادها إلى دالة أخرى أو خاصية معينة، تستخدم دوال الفيجوال بيسك العديد من المعاملات Arguments ويفصل بين كل معامل والآخر فاصلة ، حيث تعمل هذه المعاملات على منح المستخدم خيارات أكثر

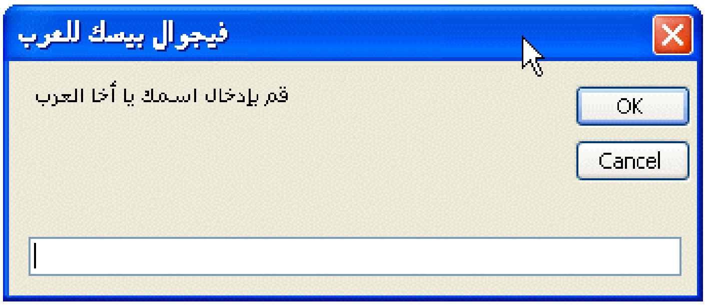
فمثلاً الدالة InputBox تكتب كالتالي:

```
FullName = InputBox(Prompt, Title)
```

فــ Prompt تعني النص الموجود أعلى صندوق المدخلات، وTitle تعني عنوان صندوق المدخلات. فإذا بدلنا بدل الكلمتين أعلاه هذا الكود:

```
FullName = InputBox("قم بإدخال اسمك يا أخا العرب"، "فيجوال بيسك للعرب")
```

سيظهر لنا صندوق المدخلات كما في الشكل التالي:



استخدام المتغيرات لعرض المخرجات:

تستطيع استعراض محتويات متغير ما بواسطة إسناد قيمة المتغير إلى خاصية ما Property مثل إسناد متغير ذو قيمة نصية إلى الخاصية Text التابعة لــ Label أو TextBox أو بواسطة تمرير المتغير كمعامل إلى صندوق حوار. من ضمن نوافذ الحوار هذه صناديق الحوار أو دوال الــ MsgBox ، عندما نستدعي الدالة MsgBox وقد تسمى Message Box فستقوم الدالة بإظهار صندوق حوار له العديد من الخيارات لتنفيذ الهدف الذي تريده من صندوق الحوار. كما قلنا إن لكل دالة العديد من المعاملات الذين يحددون كيفية التعامل معها وطبقنا ذلك على الدالة InputBox فكذلك الحال للدالة MsgBox هناك العديد من المعاملات التي تتحكم في طريقة عملها، فالشكل العام للدالة يكون بالشكل التالي:

```
MsgBox(Prompt, Buttons, Title)
```

فكلمة Prompt تعني النص الذي سيتم إظهاره للمستخدم، و Buttons هي الأزرار التي سيتم إظهارها للمستخدم هل نعم ولا أو نعم ولا وإلغاء الأمر ، وهل إلغاء الأمر وإعادة المحاولة، أما الكلمة Title فتعني عنوان صندوق الحوار. لنفترض أننا قمنا بكتابة الأمر التالي:

```
MsgBox(" خطوة بخطوة 2008كتاب فيجوال بيسك ", MsgBoxStyle.RetryCancel, _
    "فيجوال_ بيسك للعرب")
```

ووضعنا الكود تحت الحدث Click التابع لزر معين وضغطنا الزر ستظهر لنا هذه الرسالة:



للعلم هناك العديد من المعاملات Arguments التابعة للدالة MsgBox يمكن معرفتها من تعليمات بيئة التطوير الـــ Documentation، وإذا أردت معرفة هذه المعاملات Arguments وقت كتابة البرنامج بدون فتح التعليمات فيمكنك قراءة النصوص الانجليزية التي تظهر في مربع اصفر بداخل منطقة الكود بعد كتابة الكلمة MsgBox انظر الشكل التالي:



يقدم فيجوال بيسك **2008** نوعين من صناديق الحوار وهما الدالة **MsgBox** والكلاس **Message Box** لعرض النصوص ضمن نافذة صناديق الحوار، الكلاس **Message Box** توجد ضمن مجال الأسماء (أو فضاء الأسماء) **System.Windows.Forms** ومعاملاتها **Arguments** نفس المعاملات التابعة للـ **MsgBox** ونستطيع إظهارها بالطريقة **Show** وباختصار **MsgBox** تساوي **Message Box**.

ألان سنأخذ مثال تطبيقي وليكن المثال السابق مع بعض التعديلات، لنظهر البيانات التي قام المستخدم بإدخالها في صندوق الإدخال على نافذة حوار MsgBox . نفتح المثال السابق ونذهب إلى السطر الأخير : `Label1.Text = FullName` والذي يقوم بعرض المدخلات في Label1 ونحذف هذا السطر ونقوم بتغييره إلى التالي (ليقوم بعرض المدخلات في صندوق حوار):

```
MsgBox(FullName, , "نتيجة المدخلات")
```

بعد تشغيل البرنامج وإدخال الاسم سيقوم البرنامج بعرض الاسم في صندوق حوار. المثال السابق موجود في المرفقات برقم المرفق 012 .

العمل مع أنواع محددة من البيانات:

قد تصميم برنامج معين ستحتاج لبيانات محددة، فمثلاً عند تصميم برنامج لمدرسة ستحتاج برنامج لتسجيل أسماء الطلاب وتواريخ ميلادهم وأرقام شهادات الميلاد ونتائجهم في السنوات السابقة وساعات الدراسة وبعض البيانات الأخرى، لذلك فأنواع البيانات التي قد تحتاجها في مثل هذا البرنامج هي من النوع String لأسماء الطلاب ومن النوع Date لتواريخ ميلادهم ومن النوع Single لأرقام شهادات الميلاد وكذلك لـــ نتائجهم في السنوات السابقة، أما عند تصميم برنامج لسوق البورصة فالأمر سيتغير لأننا سنهتم بالأرقام بشكل كبير جدا لان الأرقام في البورصة تعني الكثير بل سنهتم بالكسور الصغيرة جداً. في الحقيقة كل نوع من أنواع البيانات سواء كان String أو Date أو Single أو غيرها من أنواع البيانات التي سنتعرف عليها في هذا الدرس له وزن

أو حجم معين على الذاكرة فإذا استخدمنا النوع الصحيح للبيانات فإن ذلك سيوفر علينا جزء من الذاكرة خاصة في البرامج الكبيرة أو البرامج التي تقوم بعمليات حسابية معقدة، للعلم تم إضافة العديد من الأنواع البيانات في عام 2005 إلى نسخة فيجوال بيسك 2005 وذلك بسبب نزول الأجهزة الحديثة في الأسواق 64-bit Computers . هذه الأنواع هي SByte, Uinteger, UShort, ULong. SByte تقبل البيانات الموجبة والسالبة بينما بقية الثلاثة الأنواع الأخرى لا تقبل غير البيانات الموجبة فقط. دعونا ألان لنعرف أنواع البيانات التي يمكن استخدامها في فيجوال بيسك 2008 ثم سنتعرف على كيفية استخداماتها، في الجدول التالي أنواع البيانات مع حجم النوع على الذاكرة بالإضافة إلى نوع البيانات وفئتها ومثال توضيحي على كل نوع:

ملاحظة: حجم نوع البيانات على الذاكرة يقاس بالــ bits وهي حجم المساحة اللازمة لحفظ وحدة قياسية من الــ ASCII (حرف من حروف الكيبورد الانجليزية يساوي 8 bits والتي تساوي 1 byte) و الآن إلى الجدول:

نوع البيانات	الحجم	مجال البيانات	مثال
Short	16-bit	-32.768 إلى 32.767	`Dim Birds As Short` `Birds = 12500`
UShort	16-bit	0 إلى 65.535	`Dim Days As UShort` `Days = 55000`
Integer	32-bit	-2.147.483.648 إلى 2.147.483.647	`Dim Insects As Integer` `Insects = 37500000`
Uinteger	32-bit	0 إلى 4.294.967.295	`Dim Joys As UInteger` `Joys = 3000000000`
Long	64-bit	-9.223.372.036.854.775.808 إلى	`Dim WorldPop As Long` `WorldPop = 4800000004`

		9.223.372.036.854.775.807	
ULong	bit–64	إلى 0 18.446.744.073.709.551.615	`Dim Stars As ULong` `Stars = _` `1800000000000000000`
Single	bit –32 floating point	- 3.4028235E38 to 3.4028235E38	`Dim Price As Single` `Price = 899.99`
Double	bit –64 floating point	- 1.79769313486231E308 to 1.79769313486231E308	`Dim Pi As Double` `Pi = 3.1415926535`
Decimal	bit–128	0 to +/- 79,228,162,514,264,337,593 ,543,950,335(+/-7.9...E+28) أو بدون علامة الألوف كالتالي: to +/- 0 7.9228162514264337593543 950335 بالإضافة إلى 28 خانة على يمين الرقم، أضف الحرف "D" إذا أردت من الفيجوال تهيئة الرقم العشري.	`Dim Debt As Decimal` `Debt = 7600300.5D`
Byte	bit–8	صفر إلى 255 (موجب فقط)	`Dim RetKey As Byte` `RetKey = 13`
SByte	bit–8	128– إلى 127	`Dim NegVal As SByte` `NegVal = -20`
Char	bit–16	أي رمز Unicode من صفر إلى 65.535 ، أضف C لتهيئة الـ Char	`Dim UnicodeChar As Char` `UnicodeChar = " "c`
String	عادة bit–16 لكل حرف	0 إلى 2 مليار من رموز الـ Unicode (كل واحد منهم bit–16)	`Dim Dog As String` `Dog = "pointer"`
Boolean	bit–16	صحيح أو خطأ وعند الكود الصفر يعتبر خطأ وبقية القيم تعتبر صحيح.	`Dim Flag As Boolean` `Flag = True`

Date	bit-64	من 1 يناير لعام 1 للميلاد إلى 31 ديسمبر لعام 9999	`Dim Birthday As Date` `Birthday = #3/1/1963#`
Object	bit-32	أي نوع من أنواع البيانات.	`Dim MyApp As Object` `MyApp = CreateObject _` `("Word.Application")`

ليس المهم أن نحفظ الجدول السابق أعلاه عن ظهر قلب ولكن الأهم هو أن نحاول تطبيقه، فنقوم بتعريف متغيرات بالاعتماد على الجدول أعلاه، وبعد أكثر من مرة من تعريف المتغير واستخدامه سنعرف طبيعة كل متغير وأوقات استخدامه. لابد علينا من استخدام المتغير المناسب حتى لا نستهلك الذاكرة Memory، فمثلاً إذا كان عدد المباني التابعة للشركة هو المتغير فنستطيع استخدام النوع Decimal أو Double أو Single أو Object أو SByte أو Byte لكننا سنستخدم النوع الذي لا يهلك الذاكرة بشكل كبير، وعليه فسنستخدم النوع Byte لان المباني التابعة للشركة ستكون خمسة أو ستة ولن تصل بالكاد إلى 100 مبنى ولكن إذا كانت الشركة شركة عقارات (لديها العديد من المباني) فيمكننا استخدام النوع Single حيث سيتيح لنا الوصول إلى الملايين من الأرقام ويعتبر أخف على الذاكرة من النوع Decimal أو Double. بعبارة أخرى يجب علينا معرفة نوع البيانات التي سنتعامل معها ثم اختيار النوع الأفضل من الأنواع أعلاه.

مثال تطبيقي على أنواع البيانات:

تعرفنا سابقاً على أنواع البيانات وأخذنا أمثلة على كل نوع من أنواع البيانات، الآن سنقوم بتصميم تطبيق للتعريف بالمتغيرات بواسطة الكود. أنشئ مشروعا جديداً وليكن اسمه Data Types نقوم بإضافة المكونات التالية للفورم مع ضبط الخصائص:

ListBox و Button وكذلك اثنين Label

ونضبط خاصية الـ Text للـ Button على "خروج"

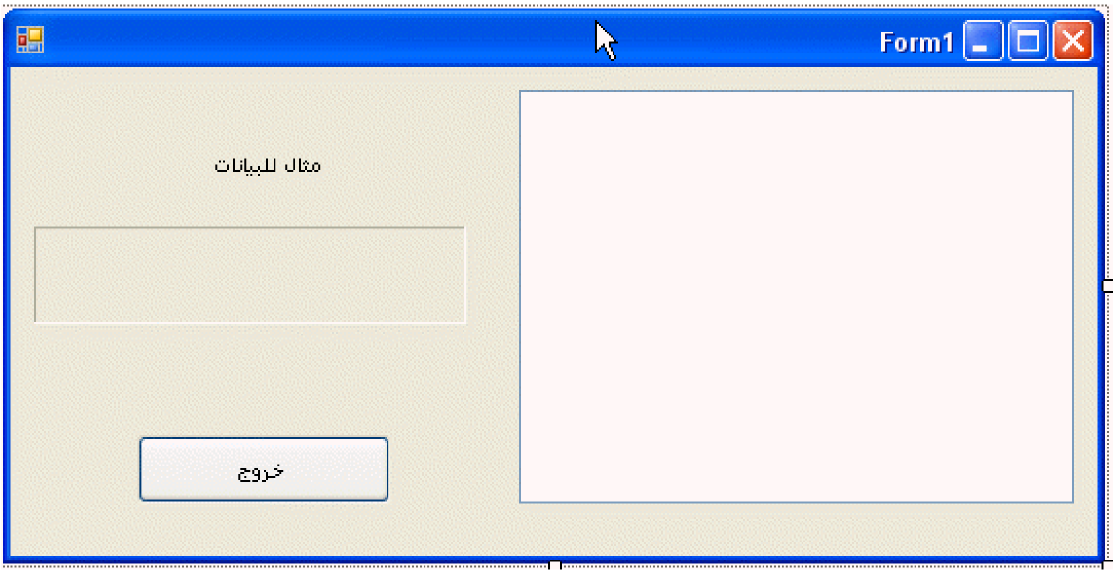
وكذلك خاصية الـ Text لأحد الـ Label على "مثال للبيانات"

أما الـــ Label الثاني فنضبط له الخصائص التالية:

BroderStyle على Fixed3D

AutoSize على False

وعلية سيكون نموذج الفورم كما في الشاشة التالية:



نقوم بتعريف المتغيرات التي سوف نستخدمها في مختلف مناطق الكود بعرفها بعد الجملة Public Class الموجودة في أعلى الفورم، بالشكل التالي:

```
Dim Birds As Short
Dim Insects As Integer
Dim WorldPop As Long
Dim Price As Single
Dim Pi As Double
Dim Debt As Decimal
Dim RetKey As Byte
Dim UniCodeChar As Char
Dim Dog As String
Dim Flag As Boolean
Dim BirthDay As Date
```

بالطريقة الموضحة أعلاه، نستطيع تعريف المتغيرات واستخدامها في كل مراحل الكود لأننا كتبنا التعريف في بداية مرحلة الكود، لكن إذا كنا نريد استخدام المتغيرات على نطاق ضيق فيمكننا كتابة التعريف في المكان المحدد الذي نريد استخدام المتغيرات فيه، مثل أن نعرف المتغيرات في الإجراء Click التابع لزر من الأزرار. أما إذا كان لدينا أكثر من فورم ونريد تعريف المتغيرات واستخدامها في كل فورم ضمن المشروع فلابد من طريقة مختلفة لتعريف المتغيرات وهي تعريف المتغيرات من النوع Public أو Global وذلك بوضع المتغيرات في ملف خاص اسمه Module يتم فيه تعريف كل المتغيرات التي سنستخدمها في أكثر من فورم. سنتعرف على التعامل مع ملفات الـــ Modules في الفصل العاشر من هذا الكتاب إن شاء الله.

نضغط Double-Click على الفورم ليفتح لنا نافذة الكود عند الإجراء Load التابع للفورم ونكتب الكود التالي لتعبئة قائمة الـــ ListBox1:

```
ListBox1.Items.Add("Short")
ListBox1.Items.Add("Integer")
ListBox1.Items.Add("Long")
ListBox1.Items.Add("Single")
ListBox1.Items.Add("Double")
ListBox1.Items.Add("Decimal")
ListBox1.Items.Add("Byte")
ListBox1.Items.Add("Char")
ListBox1.Items.Add("String")
ListBox1.Items.Add("Boolean")
ListBox1.Items.Add("Date")
```

كذلك نضغطDouble-Click على الزر خروج ونكتب كود إغلاق البرنامج:

```
End
```

وأيضاً نضغط Double-Click على الـــ ListBox1 ثم نكتب الكود التالي ليقوم بكتابة نوع البيانات التي تم اختيارها في الـــ Label2:

```
Select Case ListBox1.SelectedIndex
```

```
        Case 0
            Birds = 12500
            Label2.Text = Birds
        Case 1
            Insects = 37500000
            Label2.Text = Insects
        Case 2
            WorldPop = 6000000000
            Label2.Text = WorldPop
        Case 3
            Price = 964.84
            Label2.Text = Price
        Case 4
            Pi = 3.1415926535
            Label2.Text = Pi
        Case 5
            Debt = 7600300.5D
            Label2.Text = Debt
        Case 6
            RetKey = 13
            Label2.Text = RetKey
        Case 7
            UniCodeChar = " "c
            Label2.Text = UniCodeChar
        Case 8
            Dog = "Pointer"
            Label2.Text = Dog
        Case 9
            Flag = True
            Label2.Text = Flag
        Case 10
            BirthDay = #3/1/1980#
            Label2.Text = BirthDay
    End Select
```

التطبيق أعلاه في الملف المرفق رقم 013، سوف نتعلم الكثير من الخصائص التابعة للدالة Select Case في الفصل القادم بعون الله.

التراكيب البرمجية Structures

عندما ذكرنا أنواع المتغيرات سابقاً، قد يتساءل البعض هل هذه كل أنواع المتغيرات بعبارة أخرى هل توجد أنواع أخرى للمتغيرات، نعم هنالك متغيرات تسمى تراكيب Structures يتم تعريفها من قبل المبرمج نفسه وتسمى User Defined Types UDT أي أنواع المتغيرات المحددة عن طريق المبرمج. وتستخدم هذه الطريقة إذا كان لديك مجموعة من البيانات المترابطة فيما بينها ولكن كل نوع من أنواع مختلف عن الآخر مع وجود الرابط بين البيانات ككل. فمثلاً إذا كان لدينا

بيانات العاملين في المنشأة (اسم الموظف، تاريخ ميلاده، تاريخ التوظيف) وتستخدم هذه البيانات أكثر من مرة. فنستطيع تعريف المتغيرات بشكل جماعي على شكل تركيب Structure. ولنفهم كيفية تعريف الــ Structure علينا أن نعرف بيانات العاملين بالطريقة القديمة كالتالي:

```
Dim EmployeeName as String

Dim EmployeeDateOfBirth as Date
Dim EmployeeHireDate as Date
```

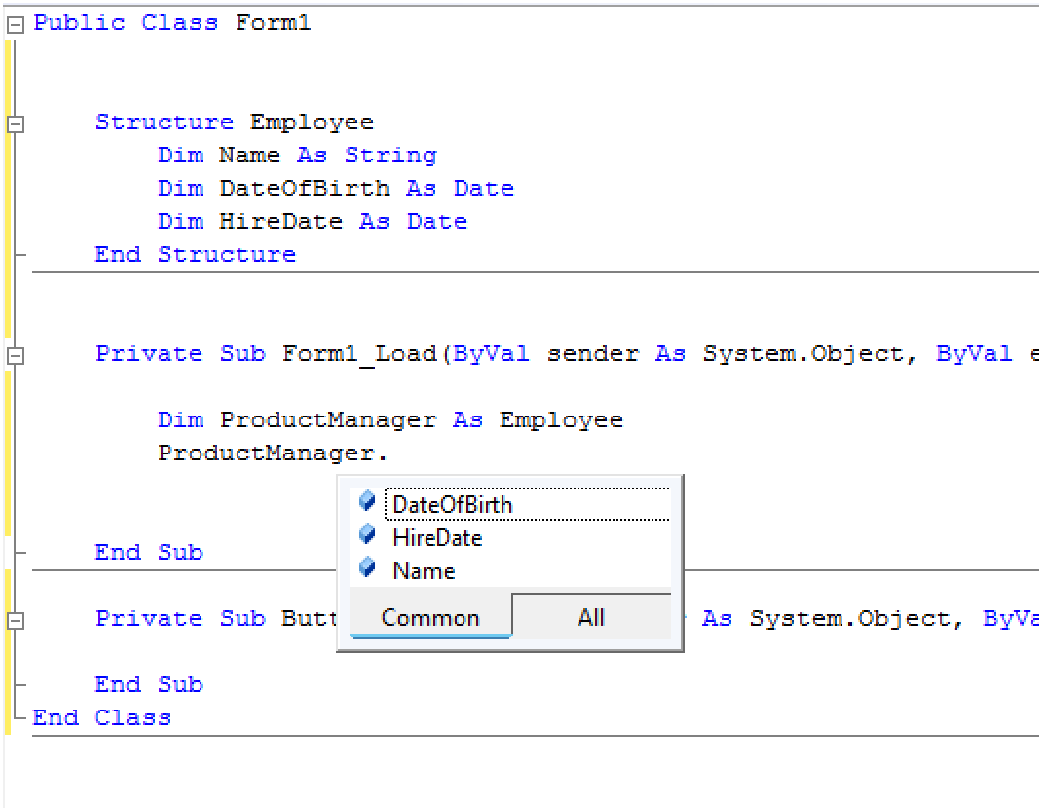
السطر الأول يقوم بتعريف اسم الموظف، الثاني لتاريخ ميلاد الموظف، الثالث لتاريخ تعيين الموظف. ولابد علينا أن نكرر الأكواد أعلاه مع كل عملية تعريف موظف جديد. أما في حالة استخدام التراكيب فإننا سنقوم بتعريف التركيب لمجموعة الموظفين مرة واحده فقط كالتالي:

```
Structure Employee
   Dim Name As String
   Dim DateOfBirth As Date
   Dim HireDate As Date
End Structure
```

لاحظ أن لدينا مجموعة من البيانات (أكثر من نوع من أنواع البيانات) ولكن هناك رابط ما بين هذه البيانات (لها علاقة بالموظفين) لذلك استخدمنا التراكيب. ثم إذا رادنا تعريف موظف جديد ضمن الكود بعد تعريف التركيب أعلاه، نكتب الكود كالتالي:

```
Dim ProductManager as Employee
ProductManager.Name = "أحمد"
```

لاحظ الصورة التالية وانظر كيف تقدم بيئة التطوير أنواع البيانات بسبب استخدامنا للتركيب:



ملاحظة أخرى حول التراكيب Structures، عندما نعرف تركيب Structure لا بد أن نقوم بتعريفه في خانه مستقلة في منطقة الكود و لا يمكن أن نقوم بتعريفه بين Private Sub و End Sub (لاحظ الصورة أعلاه).

إذا دققنا في صيغة التراكيب Structure سنلاحظ تشابهها الشديد مع الخصائص، مما يوحي لنا بأن مايكروسوفت استخدمت نفس طريقة التراكيب لتعريف خصائص المكونات.

الثوابت Constants:

تذكر معنا ماذا درسنا في الدروس السابقة، لقد درسنا المتغيرات وهي البيانات التي تتغير من وقت لآخر، أما الآن سنتعرف على الثوابت (عكس المتغيرات) ويمكن تعريف الثوابت بأنها المتغيرات التي لا تتغير قيمتها. فمثلاً قيمة المتغير π أو ما يسمى بـــ"فآي" فيعتبر من الثوابت لأن قيمته تساوي 3.14159265 فبدلاً من حفظه كمتغير في البرنامج يمكن حفظه كثابت Constant.

أعتقد الآن بأننا فهمنا معنى ثابت Constant (وهو القيمة التي لا تتغير)، قد نتساءل لماذا نستخدم

الثوابت والإجابة هي عندما يوجد لدينا قيمة لا تتغير ولتحسين أداء برنامجنا ولتسريع تعامله مع الكود، وفي نفس الوقت لتجنب الأخطاء. ولتعريف ثابت ما نستخدم الكلمة Const كالتالي:

```
Const Pi As Double = 3.14159265
```

الكود أعلاه يقوم بحفظ قيمة الثابت π أو ما يسمى بـــ"فآي".

إذا أردنا استخدام على طول الفورم فنقوم بتعريف الثابت في أعلى منطقة الفورم التابعة للفورم أما إذا أردنا استخدام الثابت في حالة واحدة فقط فنقوم بتعريف الثابت في داخل الإجراء الذي نريد أن نستخدم فيه الثابت، وفي حالة كان لدينا أكثر من فورم ونريد استخدام الثابت في كل فورم نقوم بتعريف الثابت في قالب برمجي يسمى Module مسبوقاً بالكلمة Public كما في المثال التالي:

```
Public Const pi As Double = 3.14159265
```

نستفيد من الثوابت في العديد من المهام لعل أبرزها تكوين وحل المعادلات الرياضية.

المعاملات الرياضية Operators:

ونقصد بها العلامات الرياضية مثل علامة الطرح – أو الجمع + وغيرها، وهذه هي العلامات الرياضية التي ممكن استخدامها بداخل الفيجوال بيسك 2008:

المعامل	الوصف	مثال
+	الجمع	9 = 5 + 4
–	الطرح	1 = 4 – 5
*	الضرب	20 = 5 * 4
/	القسمة	1.25 = 4 / 5
\	القسمة بدون إظهار الكسور في النتيجة	3 = 4 \ 15

Mod	باقي القسمة	`18 Mod 4 = 2`
^	الأُس	`4 ^ 3 = 64`
&	لضم الكلمات مع بعضها	`4 & 5 = 45`

إذا لاحظت المعاملات أعلاه ستجد أن هناك ثمانية معاملات، أربعة منها بسيطة (الأربعة الأولى)، وأربعة منها متقدمة وهي الأربعة الأخيرة لأننا لا نستخدمها من وقت لآخر.

مثال المعاملات الرياضية البسيطة: + ، - ، * ، /

المعاملات التابعة للجمع أو الطرح أو الضرب أو القسمة هي معاملات رياضية بسيطة التعامل وقد تعرفنا عليها في دراستنا الابتدائية في المدرسة، هنا سنقوم فقط بأخذ مثال برمجي تطبيقي لنرى كيف نستفيد من هذه المعاملات.

قم بإنشاء مشروع جديد وليكن اسمه Basic Math ثم أضف المكونات التالية للفورم:

١- عدد ثلاثة صناديق نص TextBox

٢- واحد صندوق المجموعات GroupBox

٣- أربعة أزرار راديو Radio Button إلى صندوق المجموعات

٤- ثلاثة ليبلات Labels

قم بضبط الخصائص كالتالي:

للفورم:

RightToLeft = Yes

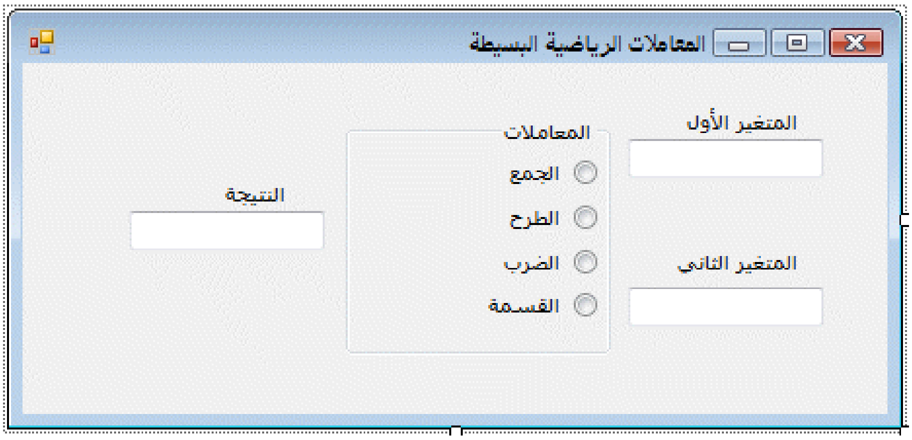
Text = المعاملات الرياضية البسيطة

للبيلات:

نغير خاصية الـــ Text للأول "المتغير الأول"، وللثاني "المتغير الثاني"، وللثالث "النتيجة"

ونغير خاصية الـــ Text لصندوق المجموعات إلى المعاملات.

وكذلك نغير خاصية الـــ Text لأزرار الراديو كالتالي: الأول "الجمع"، الثاني "الطرح"، الثالث "الضرب"، الرابع "القسمة". بعد التصميم وتعديل الخصائص سيكون الفورم مثل الشكل التالي:



نضغط Double-Click على زر الراديو "الجمع" لنذهب إلى منطقة الكود لنكتب الأكواد التالية:

```
Dim x, y As Double
X = TextBox1.Text
Y = TextBox2.Text
TextBox3.Text = x + y
```

وكذلك نكتب الكود التالي لزر الراديو الطرح:

```
Dim x, y As Double
x = TextBox1.Text
y = TextBox2.Text
TextBox3.Text = x - y
```

وهذا للضرب:

```
Dim x, y As Double
x = TextBox1.Text
y = TextBox2.Text
TextBox3.Text = x * y
```

وهذا للقسمة:

```
Dim x, y As Double
x = TextBox1.Text
y = TextBox2.Text
TextBox3.Text = x / y
```

نقوم بتشغيل البرنامج ثم نقوم بإضافة أرقام إلى صندوق النص الأول والى الثاني ثم جرب العمليات الحسابية (الجمع والطرح والضرب والقسمة). بالمناسبة نسخة من المشروع بالمرفق رقم 014.

مثال لاستخدام المعاملات الرياضية المتقدمة:

سنقوم باستخدام المثال الأول (في المرفق رقم 014) مع تغيير خاصية الـــ Text لأزرار الراديو كالتالي:

الزر الأول من "الجمع" إلى "القسمة بدون إظهار الكسور"

الزر الثاني من "الطرح" إلى "باقي القسمة"

الزر الثالث من "الضرب" إلى "الأُس"

الزر الرابع من "القسمة" إلى "ضم الكلمات"

ثم نقوم بتغيير الأكواد التابعة لكل زر فنبدل المعاملات التالية فقط:

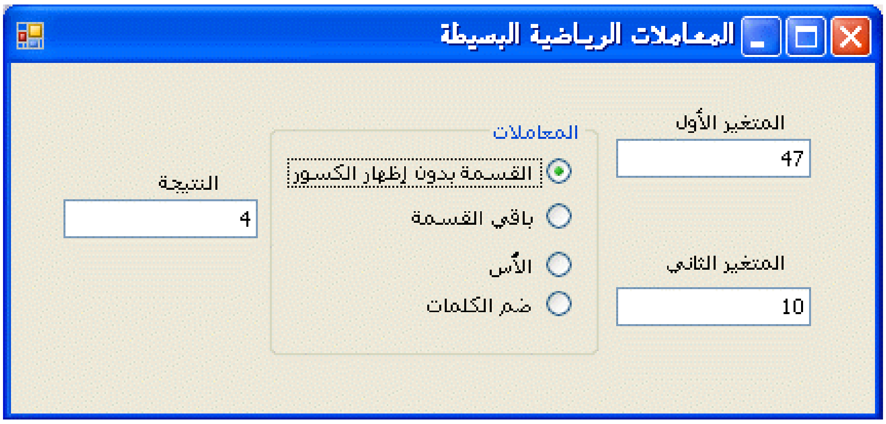
من المعامل + إلى \

من المعامل – إلى Mod

من المعامل * إلى ^

من المعامل / إلى &

بعد التعديلات السابقة، قم بتشغيل البرنامج سيكون كما في الشاشة التالية:



لاحظ أن "القسمة بدون إظهار الكسور" للرقم 47 على الرقم 10 يساوي 4 كما في الصورة أعلاه، وتفيدنا هذه العملية في حساب الأشياء التي لا تقبل التقسيم إلى مكونات اصغر فمثلا يمكن حساب عدد الأشخاص الذين تستطيع حملهم السيارة بواسطة المعامل أعلاه. (~~فلا يمكن أن نقول حمولة السيارة = 4.7 أشخاص~~)، وكذلك عدد السيارات التي قد تحملها السفينة الواحدة، وغيرها من العمليات التي لا تقبل التقسيم إلى مكونات اصغر. للعلم التطبيق أعلاه موجود بالمرفقات رقم 015.

التعامل مع الطرق Methods ضمن بيئة التطوير:

من خلال العمل في بيئة التطوير قد تحتاج العديد من الأوامر لصنع معادلات مختلفة أو لتصميم برنامج يحسب مجموعة من القيم أو المتغيرات، فالطرق المقصودة هنا هي تلك المعادلات المخزونة مسبقاً ضمن بيئة التطوير وبالتحديد ضمن الفريم وورك 3.5 (أي يمكن أن نستخدمها في الفيجوال بيسك أو في السي شارب أو في بقية برامج بيئة التطوير فيجوال 2008). لمراجعة جميع هذه الطرق يمكنك الرجوع لها في الكلاس System.Math:. انظر الجدول أدناه لبعض الدوال أو الطرق المتوفرة ضمن الكلاس System.Math، وللعلم إذا أردت استخدام أياً من هذه الطرق أو غيرها من التابعة للكلاس System.Math يجب أن نقوم باستيراد الكلاس System.Math

ضمن مجالات الأسماء في أعلى منطقة الكود قبل كتابة أي كود آخر (حتى قبل Form1 Public Class):

الطريقة	الوصف
n ترمز إلى المتغير (بحيث يمكن استبدال n بأي رقم) (وحدة قياس الزوايا المعتمدة هي "الراديان")	
Abs(n)	ترجع لنا قيمة n بالموجب بدون سالب إذا كان سالب، وعدم تغييره إذا كان موجب.
Atan(n)	لمعرفة ظتا الزاوية n، بوحدة قياس "الراديان"، وتساوي tan أُس -1.
Cos(n)	لمعرفة جتا (جيب تمام) الزاوية أو n.
Exp(n)	الدالة الأُسية: وتعتبر n هي الأُس بالنسبة للأساس 2.718281828.
Sign(n)	ترجع لنا القيمة -1 إذا كانت n أصغر من صفر وترجع القيمة صفر إذا كانت n تساوي صفر وترجع لنا القيمة +1 إذا كانت n أكبر من الصفر.
Sin(n)	ترجع لنا جيب الزاوية n.
Sqrt(n)	تظهر لنا الجذر التربيعي لـــ n.
Tan(n)	لمعرفة ظل الزاوية n، بوحدة قياس "الراديان"

سنقوم الآن بتصميم تطبيق بسيط يقوم بحساب الجذر التربيعي لرقم ما بالاستفادة من الكلاس System.Math وبالتحديد من الطريقة Sqrt(n):

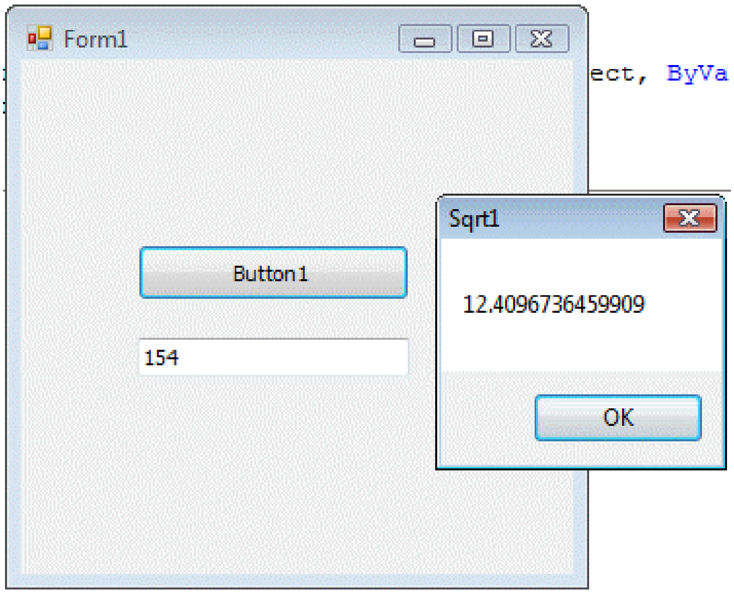
ننشئ مشروع جديد ونسميه Sqrt1 ونضيف للفورم واحد TextBox وزر Button. ثم نكتب الكود التالي في حدث Click التابع للزر Button1:

```
MsgBox(Sqrt(TextBox1.Text))
```

ونكتب الكود التالي في أعلى الفورم لاستيراد مجال الأسماء:

```
Imports System.Math
```

بعد تشغيل البرنامج نكتب الرقم المطلوب إيجاد الجذر التربيعي له في مربع النص ثم نضغط الزر، سيقوم البرنامج بإظهار الجذر التربيعي له في رسالة الحوار Msg Box. كما في الشكل التالي:



المثال أعلاه في المرفق رقم 016.

ترتيب العمليات الحسابية في بيئة التطوير:

لنفرض أن لدينا العملية التالية:

10 + 15 * 2 / 4 ^ 2

فكم يكون الناتج بعدما تقوم بيئة التطوير بحساب العملية، من أجل معرفة الناتج يجب أن نعرف كيف يقوم فيجوال بيسك بسلسلة العملية، يجب أن نعرف أن هناك خطوات يقوم بها فيجوال بيسك لمعرفة الناتج و هذه الخطوات الرياضية كالتالي:

المعامل **الشرح**

١- () يقوم البرنامج بحساب ما بين الأقواس أولاً.

٢- ^ يقوم البرنامج بحساب الأُس ثانياً.

٣- - يقوم البرنامج بحساب الأرقام السالبة ثالثاً.

٤- */ يقوم البرنامج بحساب الضرب والقسمة رابعاً.

٥- \ يقوم البرنامج بعملية القسمة بدون احتساب كسور في النتيجة.

٦- Mod يقوم البرنامج باحتساب باقي القسمة سادساً.

٧- +- وفي الأخير يقوم البرنامج بعمليتي الجمع والطرح.

إذن لنرى العملية السابقة ونرى طريقة تسلسل حلها:

10 + 15 * 2 / 4 ^ 2

10 + 15 * 2 / 16

10 + 30 / 16

10 + 1.875

11.875

استخدام الأقواس لترتيب تسلسل العمليات الحسابية:

لنفهم ما علاقة الأقواس بترتيب تسلسل العمليات الحسابية نأخذ هذا المثال:

(8 - 5 * 3)^2

ناتج العملية أعلاه سيكون **49**، فيما إذا أضفنا قوسين للعملية كالتالي:

()8 - 5(* 3)^2

سيكون الناتج للعملية يساوي **81.** فمن المثال أعلاه يتضح لنا أهمية استخدام الأقواس لتنظيم وترتيب تسلسل تنفيذ العمليات الحسابية.

خلاصة الفصل الخامس:

من اجل أن	قم بالتالي
تعرف متغير	اكتب الكلمة Dim ثم اكتب بعدها اسم المتغير بعدها As وبعدها نوع البيانات الذي يمثله ذلك المتغير. لتستخدم المتغير في جميع الإجراءات في الفورم ولا تريد إعادة تعريفه أكثر من مرة، قم بتعريفه في بداية الكود وقبل أي كود تابع لإجراء في الفورم. مثال على تعريف المتغير: Dim Country as String
تعيين قيمة للمتغير	نستخدم الـــ = لتعيين قيمة معينة لمتغير محدد كالتالي: Country = “Japan”
الحصول على المدخلات من المستخدم	نستخدم الدالة InputBox و نسند القيمة إلى متغير كالتالي: UserName = InputBox("What is your name?")
إظهار المخرجات في صناديق حوار	نستخدم الدالة MsgBox كما في المثال التالي: MsgBox(“Report Kind”, , "Spain Weather Report")
تعريف ثابت	نكتب Const وبعدها اسم الثابت ثم = بعدها القيمة الثابتة كالتالي: Const AhmedAge as Short = 22
كتابة عملية حسابية	نستخدم الإشارات الرياضية السبعة المذكورة للربط بين الأرقام في عملية

	حسابية ثم نسند النتيجة إلى متغير أو خاصية معينة كالتالي: Result = 1 ^ 2 * 3 \ 4
ادمج كلمتين أو نصين	نستخدم أداة الربط بين النصوص وهي & كالتالي: Msg = "Hello" & "," & " world!"
استيراد مجالات الأسماء	لاستيراد مجال معين من مجالات الأسماء نكتب في أعلى منطقة الكود وقبل كل شيء الكلمة Import ثم نتبعها بنوع الكلاس التي نريدها كالتالي: Imports System.Math
استدعاء طريقة Method	لاستدعاء طريقة Method لا بد أولا من استيراد مجال الأسماء الخاص بالطريقة ثم نكتب الطريقة Method المرادة فغي الكود كالتالي: فبعد استيراد مجال الأسماء : System.Math نكتب الكود التالي لاستدعى الطريقة Method المدعومة من قبل مجال الأسماء وهي Sqrt كالتالي: Hypotenuse = Sqrt(x ^ 2 + y ^ 2)
التحكم بتسلسل العمليات الحسابية ضمن الكود	يمكننا التحكم بتسلسل العمليات الحسابية ضمن الكود بإضافة أقواس ليقوم البرنامج بحساب ما بداخل الأقواس أولاً. فمثلاً: Result = 1 + 2 ^ 3 \ 4 = النتيجة تساوي3 Result = (1 + 2) ^ (3 \ 4) = النتيجة تساوي1

الجزء الثاني: أساسيات البرمجة (من الفصل الخامس إلى الفصل الثالث عشر)

الفصل الخامس: جُمل القرارات Decision Structures

كثيرا ما نحتاج في البرمجة إلى جُمل فيها قرارات معينة، فمثلاً يمكننا كتابة جملة شرطية في البرمجة باستخدام أداة الشرط إذا If ، فنقول فيها إذا كان أ أكبر من ب فنفذ الأمر التالي ونكتب أمر معين أما إذا كان ب أكبر من أ فنفذ الأمر التالي ونكتب أمر معين آخر، وتسمى هذه الجمل بالجمل الشرطية، وهناك جمل أُخرى لها علاقة باتخاذ قرارات معينة وقت تشغيل البرنامج. هذه الجمل وغيرها سنتعرف عليها في هذا الفصل.

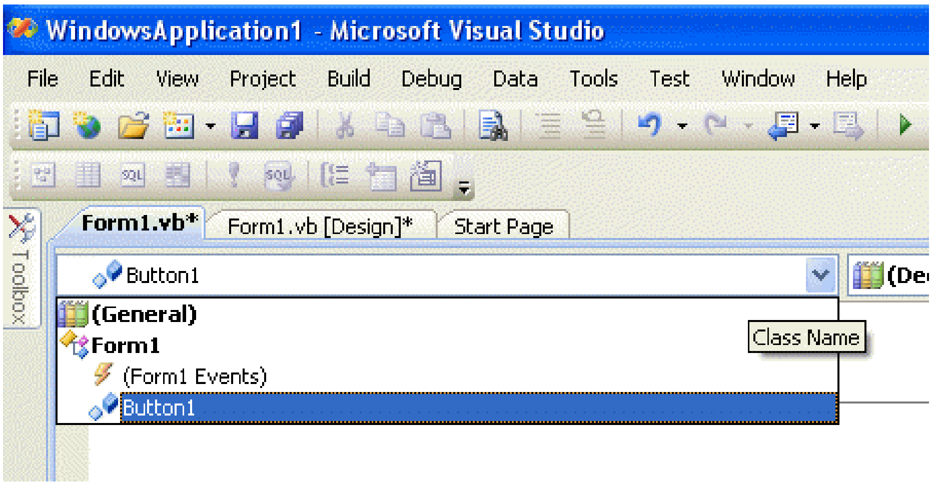
البرمجة بالأحداث Event-Driven Programming:

في التطبيقات التي قمنا بتصميمها سابقاً لم نشر إلى مفهوم البرمجة بالأحداث، وذلك لان التطبيقات التي تناولناها سابقا كانت عبارة عن تطبيقات تتعامل مع الأدوات والقوائم المتوفرة ضمن بيئة التطوير. فالبرمجة بالأحداث Event-Driven Programming هي عبارة عن برنامج مكون من مجموعة من الكائنات الذكية Objects تستقبل المدخلات من المستخدم ثم تقوم بمعالجتها ثم تنتظر الاختيار من المستخدم. قد تكون هذه المدخلات من قبل المستخدم أو من قبل الكمبيوتر بدون تدخل من المستخدم فيمكن جعل تاريخ الجهاز عبارة عن مدخل من المدخلات بدون تدخل من المستخدم أو ممكن ينتظر الكمبيوتر لحين تحقق شرط ما مثل وصول رسالة بالإيميل أو بالفاكس ثم تنفيذ أمر معين. تذكر سابقاً (في التطبيقات التي تعاملنا معها) كنا نستخدم الحدث Click لزر الـــ Button أو CheckedChanged لصندوق التأشير CheckBox أو SelectedIndexChanged التابعة لكائن القائمة، وهناك العديد من الأحداث التابعة لكل مكون من مكونات Objects بيئة التطوير. **فالبرمجة بالأحداث هي البرمجة التي تعتمد بشكل أساسي على الأحداث Events التابعة للمكونات المتوفرة في بيئة التطوير**، فبدلاً من كتابة كود من صفحتين أو ثلاث لمراقبة المستخدم حتى يضغط على الزر Button فقط نكتب الأمر الذي نريد تنفيذه في الحدث Click التابع للزر Button، فالبرمجة بالأحداث توفر علينا الكثير من الجهد والوقت بسبب وجود العديد من العديد المخزونة مسبقاً في بيئة التطوير.

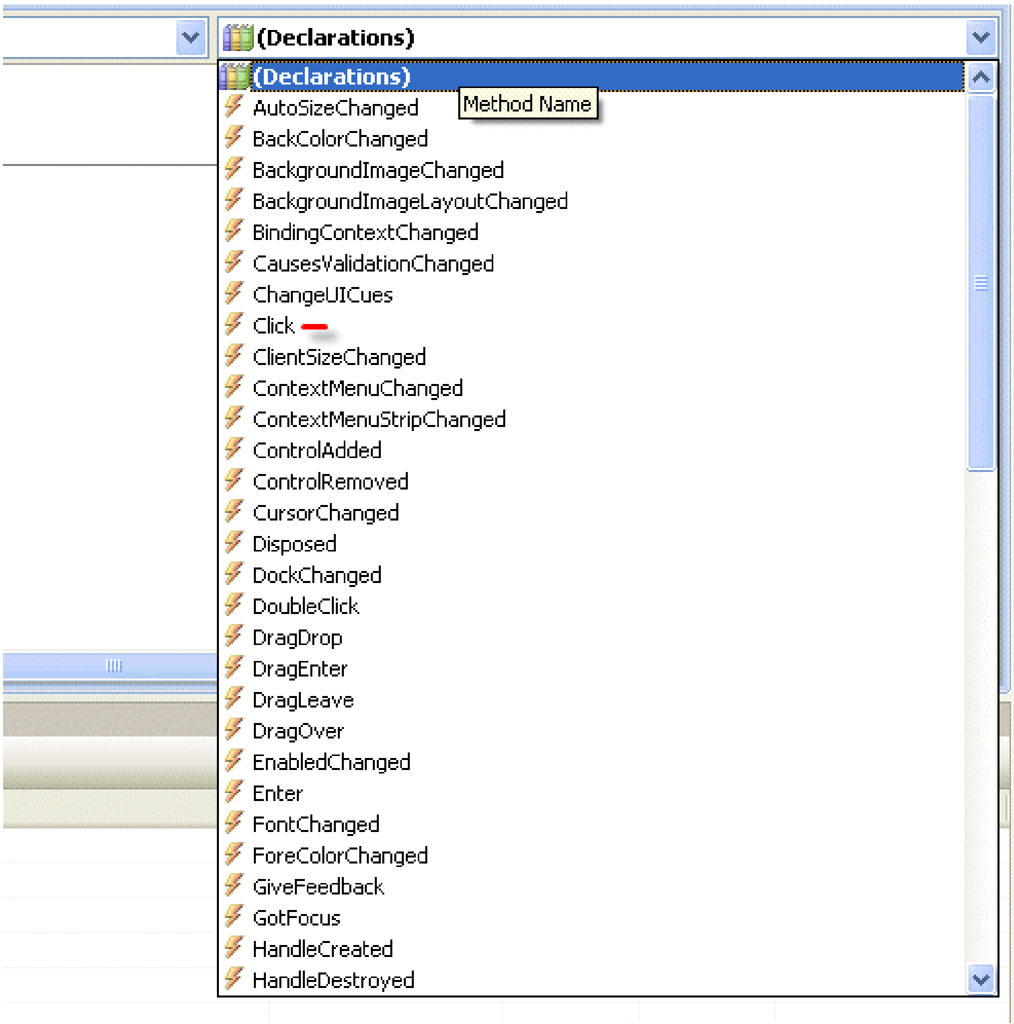
في هذا الفصل سنتعلم كيفية استخدام جُمل القرارات Decision Structures للمقارنة بين المتغيرات والخصائص والقيم ثم اتخاذ قرارات محددة في حالة تحقق شروط معينة.

الأحداث المدعومة من الفيجوال بيسك:

هناك الكثير من الأحداث المدعومة من الفيجوال بيسك لكل كائن Object من الكائنات في بيئة التطوير، لمعرفة الأحداث المدعومة لكل كائن فقط نقوم بإضافة الكائن إلى الفورم ثم نفتح نافذة الكود لنرى الأحداث المدعومة، لننشئ تطبيقاً جديداً ونضيف للفورم الكائن Button ثم نفتح نافذة الكود ونذهب إلى أعلى يسار الصفحة ونفتح القائمة المنسدلة التي تظهر كما في الشكل التالي:



لاحظ هنا وجود اثنين من المكونات فقط وهما Button1 و Form1 (في الحقيقة ستجد في الصورة كل المكونات المتوفرة على الفورم ولأنه لا يوجد لدينا إلا اثنين من المكونات فقط فقد ظهرت في القائمة وإذا كان هناك أكثر من مائة مكون لظهرت جميعها) فإذا أردنا اختيار الأحداث التابعة للفورم1 نختار Form1 Events وإذا أردنا اختيار الأحداث التابعة للكائن Button1 نختار Button1 من القائمة ثم نذهب إلى القائمة المنسدلة المتواجدة على اليمين لنختر منها الحدث الذي نريده والتي ستكون كالشكل التالي:



في القائمة المنسدلة ستجد كل الأحداث التي تحتاجها أو التي قد تحتاجها في حياتك، وقد تحتاج نادراً جداً جداً بعض الأحداث الغير متوفرة أعلاه في حينه لابد عليك من تصميم كود للحدث المراد، لاحظ أننا قد استخدمنا 16 مثال حتى الآن في هذا الكتاب ولم نستخدم إلا واحد من الأحداث أعلاه وهو الحدث Click (الذي أمامه خط أحمر صغير في الصورة)، وبقية الأحداث قد تحتاج لبعض منها خلال حياتك البرمجية.

استخدام الجمل الشرطية:

الجمل الشرطية هي جزء من البناء البرمجي للجملة البرمجية وتسأل هذه الجملة عن خاصية معينة أو قيمة وعينة لتخرج لنا أحد الخيارين (صح أو خطأ) كما في المثال التالي:

Price < 100

ففي الجملة السابقة حددنا أن السعر Price أقل من 100، فنحن الآن أمام واحد من خيارين إما أن يكون السعر فعلاً أقل من مائة وتكون الجملة الشرطية صحيحة و أو أن يكون السعر أكثر من مائة وعلية تكون الجملة الشرطية خاطئة. قبل أن ندخل أكثر في الجمل الشرطية نتعرف على هذه المعاملات (أو الرموز):

المعامل	**معناه**
=	يساوي
<>	لا يساوي
<	أكبر من (تقرأ في سطر الأكواد من اليسار إلى اليمين)
>	أصغر من (تقرأ في سطر الأكواد من اليسار إلى اليمين)
=<	أكبر من أو تساوي (تقرأ في سطر الأكواد من اليسار إلى اليمين)
=>	أصغر من أو تساوي (تقرأ في سطر الأكواد من اليسار إلى اليمين)

أمثلة على المعاملات الشرطة أعلاه:

Price < 100

لكي تكون الجملة الشرطية أعلاه صحيحة فلا بد أن يكون السعر أقل من مائة وإذا كان أكثر فالجملة الشرطية خاطئة، ماذا إذا كان السعر يساوي مائة ستظل الجملة الشرطية خاطئة لأننا حددنا في الجملة الشرطية بأن السعر لابد وأن يكون أقل من مائة، أما إذا كانت الجملة الشرطية كالتالي:

Price <= 100

فإن السعر إذا كان مائة أو أقل فإن الجملة الشرطية صحيحة (أو بالأصح نتيجة الجملة الشرطية صحيحة). لننظر جملة شرطية أُخرى:

Tall <> 150

فإذا كان الطول Tall يساوي 150 فإن نتيجة الجملة خاطئة أما إذا كان لا يساوي 150 فإن نتيجة الجملة الشرطية صحيحة. لنرى كيف يمكن الاستفادة من هذه الشروط كالتالي:

Score < 20

Score = Label1.Text

فإذا كان الرقم الموجود في Label1 أصغر من 20 فإن نتيجة الجملة الشرطية صح وإذا كان الرقم الموجود في Label1 أكبر أو يساوي 20 أو كان عبارة عن نص فإن نتيجة الجملة الشرطية خاطئة.

جملة إذا كانت ... فـــ If Then Decision Structure

إذا كان لدينا شرط ما إذا تحقق فإن برنامجنا يقوم بتنفيذ أمر ما فإننا نستخدم أداة الشرط **إذا** كالتالي

إذا كان (الشرط) فقم بالتالي

If *condition* then *statement*

مثال:

If Score >= 20 Label1.Text = "You Win"

فإذا كانت الدرجة Score تساوي 20 أو أكثر (فقط) ستظهر الجملة "You Win" في Label1، أي سيقوم فيجوال بيسك بتغيير الخاصية Text التابعة للمكون Label1 إلى الجملة الموضحة غير ذلك لن يتم تغيير الخاصية Text للمكون المذكور.

جملة If الشرطية التي تعالج أكثر من شرط:

نستطيع معالجة أكثر من شرط باستخدام أداة الشرط If، لدينا المثال التالي:

```
If Result < 50 Then
    MsgBox("راسب")
ElseIf Result >= 50 Then
    MsgBox("ناجح")
```

كما هو واضح في المثال أعلاه بأننا نستخدم ElseIf لإضافة شرط جديد إلى الجمل الشرطية، فإذا كانت النتيجة أقل من 50 درجة فإن الطالب راسب وإذا كانت أكثر من خمسين درجة فإن الطالب ناجح ويكون تنفيذ الأمر بإظهار صندوق حوار Message Box فيه كلمة ناجح أو راسب. ماذا إذا كنا نريد من التطبيق إظهار تقدير للطالب(مقبول، جيد، جيد جداً، ممتاز)، في مثل هذه الحالة سنحتاج للكثير من الجمل الشرطية المترابطة فيما بينها كما في المثال التالي:

```
Dim Result As Single = 71
If Result < 50 Then
    MsgBox("راسب")
ElseIf Result <= 60 Then
    MsgBox("مقبول")
ElseIf Result <= 70 Then
    MsgBox("مقبول مرتفع")
ElseIf Result <= 80 Then
    MsgBox("جيد جداً")
ElseIf Result <= 90 Then
```

```
    MsgBox("جيد جداً مرتفع")
ElseIf Result <= 100 Then
    MsgBox("ممتاز")
Else
    MsgBox("هناك خطأ ما، تأكد من المدخلات")
End If
```

عند تنفيذ البرنامج سيقوم الكمبيوتر بالتأكد من قيمة النتيجة Result فإذا كانت أصغر من 50 سيظهر صندوق حوار يفيدنا بأن الطالب راسب لكن في حالة ما درجات الطالب ناجح ولكن أصغر من 60 سيظهر صندوق حوار يفيد بأن الطالب مقبول، (للتأكد فقط يمكننا تغيير قيمة Result الموجودة في أول سطر) وهكذا مع بقية الشروط أو الحالات التي حددناها في الأسطر البرمجية أعلاه من جيد إلى ممتاز، ما يهمنا الآن هو الشرط الأخير Else وما بعدها فهذا الأمر أو هذه الحالة تعني أنه إذا لم تطابق أي حالة من الحالات أعلاه فقم بتنفيذ هذا الأمر وتفيدنا Else كثيراً إذا كان لدينا العديد من الخيارات الخاطئة وخيار واحد فقط هو الصحيح (أو العكس)، فبإمكاننا كتابة الأمر If للخيار الوحيد واستخدام Else لجميع الخيارات الأُخرى.

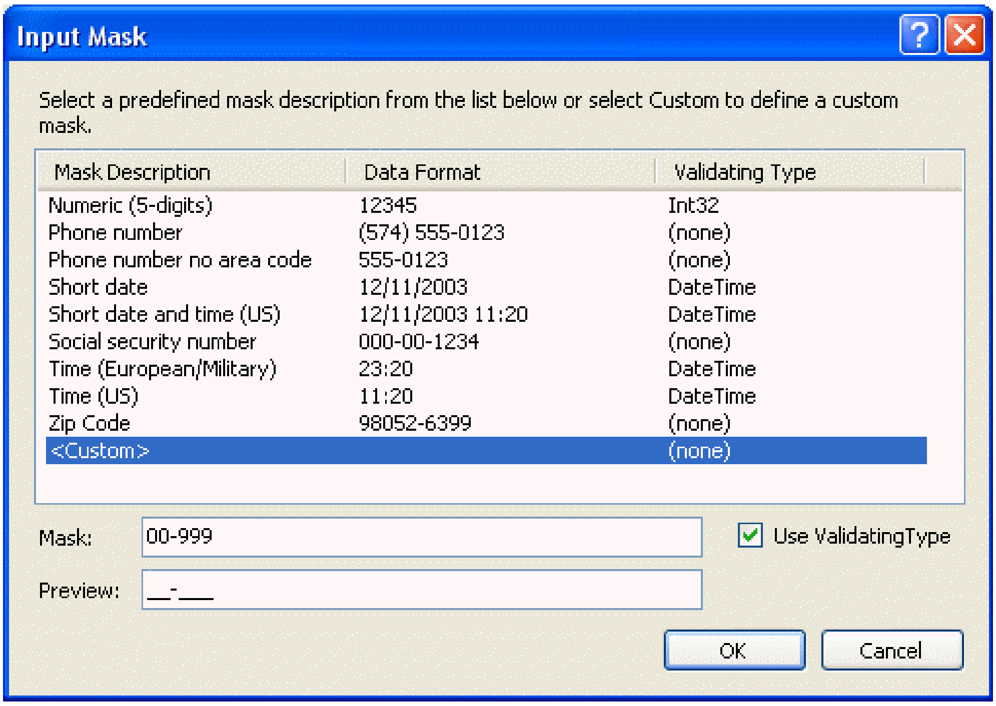
ملاحظة هامة:

الترتيب في جملة If الشرطية هام جداً، لان الكمبيوتر يتوقف عن البحث عن أقرب الشروط المطابقة عند وجود أول شرط مطابق(وعليه فلا بد أن يكون الترتيب بشكل تصاعدي للجمل الشرطية بعبارة أُخرى من الصغير إلى الكبير)، ففي المثال أعلاه إذا بدأنا بالشرط قبل الأخير (ElseIf Result <= 100 Then) سيعتمد الكمبيوتر هذا الشرط بدون الرجوع لبقية الشروط ولو كانت صحيحة فسيظهر صندوق النص "ممتاز" ولو كانت الدرجة فقط **70**، فلا بد علينا أن نتتبه للترتيب في جملة If.

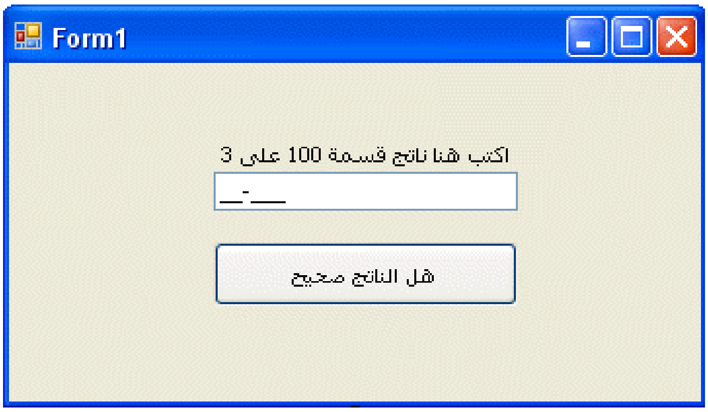
مثال تطبيقي:

لنأخذ مثال بسيط على If الشرطية، يوجد لدينا في هذا المثال خياران فقط، فيوجد خيار واحد فقط الذي صحيح وبقية الخيارات خاطئة وبسبب هذا سنستخدم If و Else كالتالي:

ننشئ مشروعاً جديدا وليكن اسمهCalculate ، نضيف للفورم MaskedTextBox و Label و Button، نستخدم هنا الـــ MaskedTextBox بدلاً عن صندوق النص **الاعتيادي بسبب توفر خاصية مميزة فيه وهي تخصيص طريقة كتابة النص بداخله** فبعد إضافة المكون MaskedTextBox إلى الفورم نضغط على المثلث الأسود في أعلى يمين المكون ثم نختار Set Mask ليظهر لنا الشكل التالي نستخدمه لتخصيص طريقة عرض النص بداخله:



نختار Custom ثم نعدل Mask كما هو موضح بالصورة. بعد إضافة المكونات و التعديل عليها يظهر الفورم كما في الشكل التالي:



نضغط Double-Click على الزر Button لندخل مباشرة إلى الحدث Click التابع للزر Button (هل تتذكر البرمجة بالأحداث!) ثم اكتب الكود التالي:

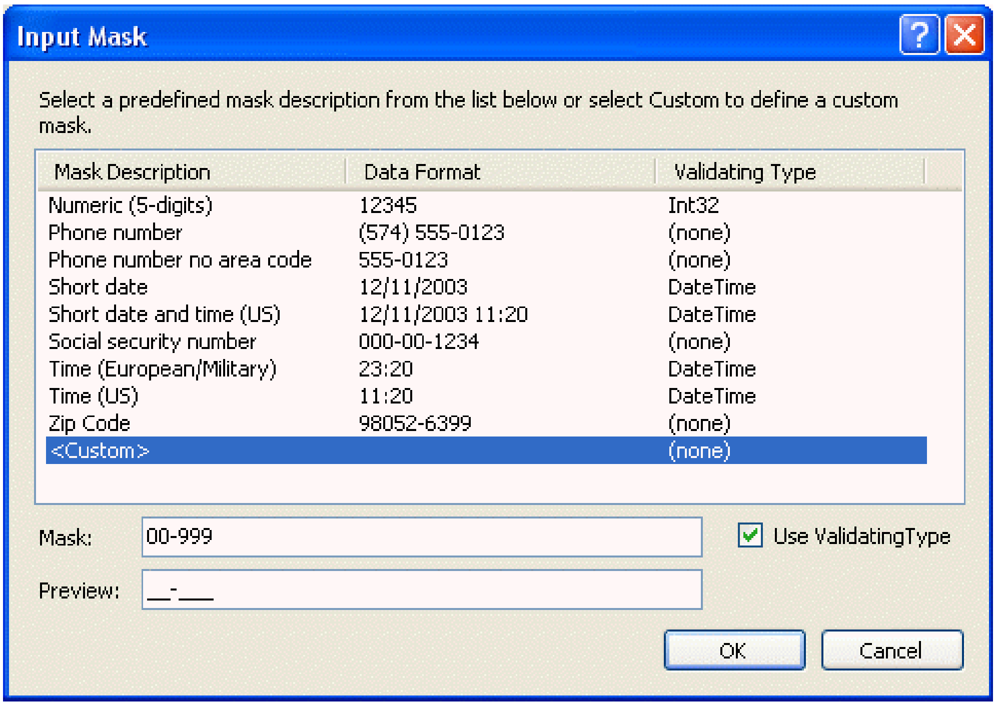
```
If MaskedTextBox1.Text = "33-333" Then
    MsgBox("الناتج صحيح")
Else
    MsgBox("الناتج خاطئ")
End If
```

قم بتشغيل البرنامج وجرب أدخل الرقم 33.333 في صندوق النص ثم اضغط على الزر سيظهر لك صندوق الحوار "الناتج صحيح"، جرب إدخال أي رقم أو نص آخر ماذا تلاحظ.

توجد نسخة من المثال أعلاه مع المرفقات برقم 017.

استخدامات الـ MaskedTextBox:

كما هو واضح في الصورة:



نستطيع استخدام الــ MaskedTextBox لإجبار المستخدم على كتابة نص محدد بطريقة معينة، فمثلا في أمريكا يكتبون رقم الهاتف الموبايل بالصيغة التالية: (574) 555-0123، في بلدي (اليمن) يفضل كتابة رقم الموبايل على ثلاث مجموعات لأن رقم الموبايل يتكون من تسع خانات كالتالي: 777 xxx xxx فباستخدام الـ MaskedTextBox يمكننا تخصيص طريقة كتابة النص داخل صندوق النص (انظر إلى صندوق النص المقابل لــ Mask في الصورة أعلاه نجد أننا نستعمل الأصفار والتسعات للتعبير عن السماح بكتابة أرقام بدلاً عن الأصفار والتسعات في غيرها من المكان الخالي في صندوق النص لا يمكن للمستخدم أن يدخل أية بيانات أخرى) فيمكننا أن نلزم المستخدم بإدخال تواريخ أو أرقام بصيغة معينة أو غيرها.

استخدام المعاملات المنطقية في الجمل الشرطية:

خلال كتابة الجمل الشرطية قد نحتاج إلى إضافة بعض الشروط في الجملة الواحدة أو قد نحتاج إلى تعقيد الشرط البرمجي في هذه الحالة لابد علينا من استخدام المعاملات المنطقية في الجمل الشرطية. دعونا نتعرف على المعاملات التالية:

١- And (وتعني واو الإضافة) إذا كان الشرطين صحيحين فالجملة الشرطية صحيحة.

٢- Or (وتعني **أو**) إذا كان أحد الشرطين صحيحاً فإن الجملة الشرطية صحيحة.

٣- Not (وتعني **ليس**) إذا كان الشرط خاطئاً فالجملة الشرطية صحيحة، وإذا كان صحيحاً فالجملة خاطئة.

٤- Xor (وتعني **إذا كان أحد**) إذا كان أحد الشرطين **فقط** صحيحاً فإن النتيجة صحيحة إذا كان الشرطين صحيحين فالجملة خاطئة.

لفهم المعاملات المنطقية دعونا نأخذ هذا المثال، لنفترض أنه لدينا اله هندسية "المنقلة" وسعرها 15 ريال. فإذا كانت لدينا هذه الشروط المنطقية:

المثال المنطقي : النتيجة المنطقية

الآلة الهندسية = المنقلة And سعرها > 20 ريال: صحيحة لأن الشرطين صحيحين

الآلة الهندسية = المسطرة Or سعرها > 30 ريال : صحيحة لأن أحد الشرطين صحيحاً

السعر ليس (Not) أكثر من عشرة : الشرط خاطئ وعلية فالجملة البرمجية صحيحة.

الآلة الهندسية = المنقلة Xor السعر > 20 ريال : خاطئة لأن كلا الشرطين صحيحين.

أعرف بأن البعض سيتعرض لبعض الصعوبات في هذه الجمل المنطقية، وخاصة الجملة الأخيرة لكن لا بأس من المعاودة في قراءة درس المعاملات المنطقية من جديد لكي نفهم أصول البرمجة، فالبرمجة ليست عبارة عن أكواد موجودة مسبقاً ضمن بيئة التطوير أو متوفرة في صفحات الانترنت، لا، إذا كنت تظن أن هذه البرمجة فأنصحك بأن تتوقف الآن والعودة بعد أن تجد في نفسك روح المثابرة والهمة العالية، سيقول البعض لماذا، أقول لك إذا كانت البرمجة عبارة عن نسخ أكواد من بيئة التطوير أو من صفحات الانترنت فأين العامل البشري أو المحدد العقلاني، ولماذا اشتهرت برامج وماتت أُخرى، إن العمليات المنطقية في البرمجة هي الأساس العملة لكل أنظمة التشغيل ولكل البرامج الكمبيوترية وكل تطبيقات الانترنت، ستقول لي وماذا تفعل معي أو معنا من أول الكتاب (كنت تعطينا أمثلة وتقول فيها أنشئ تطبيقاً جديداً ثم أضف كذا وكذا من الأدوات ثم أضف الكود الفلاني إلى الحدث الفلاني، قم بتشغيل التطبيق) سأقول لك نعم فهذه هي

مهمتي كمترجم لما جاء في الكتاب ثم لأن عملية تنظيم الكتاب جاءت بهذا الشكل لنتعرف على مزايا اللغة ثم نختار المواصلة معها أو نتوقف، أعتقد انه بعد الأمثلة التوضيحية من أول الكتاب إلى الآن لابد أن تكون قد عرفت كيف تفتح ملف تطبيق سابق وكيف تنشئ تطبيق جديد وأين تقوم بوضع الكود المعين وما معنى الحدث، أما من الآن فصاعداً فلا بد عليك أن تنتقل من طور التقليد إلى طور الإنتاج من السؤال بـــ **كيف** إلى السؤال بـــ **لماذا**، علينا الآن التعرف على أساسيات البرمجة والمنطق عند الحواسيب. عفواً على الخروج عن الموضوع ولكن نعود الآن المعاملات المنطقية، مرة أُخرى سنأخذ جمل منطقية أُخرى لفهم المعاملات المنطقية بشكل أفضل: لنفترض أن لدينا عربة (Vehicle) عبارة عن دراجة هوائية (Bike) وبسعر 200

Vehicle = "Bike" And Price < 300

ستكون نتيجة الجملة السابقة صحيحة لأن كلا الشرطين صحيحين.

Vehicle = "Car" Or Price < 500

النتيجة صحيحة لأن أحد الشرطين صحيحاً.

Not Price < 100

النتيجة صحيحة لأن عكس الشرط كان خاطئاً

Vehicle = "Bike" Xor Price < 300 False

بما أن كلا الشرطين صحيحاً، فإن النتيجة ستكون خاطئة، أما إذا كان أحدهما **فقط** (فقط أحدهما) صحيحاً فستكون النتيجة صحيحة.

ملاحظة:

عندما تستخدم العديد من المعاملات المترابطة في الجملة البرمجية فإن الكمبيوتر يقوم بالتعامل أولاً مع المعاملات الرياضية (+،− وغيرها) ثم يقوم بالتعامل مع معاملات المقارنة (<، > ، =، وغيرها)، وثالثاً يقوم بتحليل المعاملات المنطقية (**And, Or**، وغيرهما)

استخدام AndAlso و OrElse

نستطيع أن نقول أن إضافة المعاملات أعلاه إلى بيئة التطوير جاءت من قبيل زيادة الإنتاجية، أو من طريق التطوير الذي يزداد يوما بعد يوم في الفيجوال بيسك. هذين المعاملين يعتبران جديدين على مبرمجين فيجوال بيسك 6. فمثلاً المعامل المنطقي AndAlso هو نفس المعامل And ولكن مع إضافة صغيرة جداً وهي التحقق من الشرط الأول قبل الانتقال إلى الثاني فإذا كان الشرط الأول خاطئاً لا يقوم بالانتقال إلى الشرط الثاني بل يقوم مباشرة بالإبلاغ بأن نتيجة الجملة خاطئة مما يزيد من سرعة التأكد من الجمل البرمجة الشرطية بدون التحقق من جانبي الشرط، أما المعامل And فيقوم بالتأكد من الشرط الثاني ثم يظهر نتيجة الخطأ (وهذا مما لا داعي له إذا استخدمنا المعامل AndAlso). أما المعامل OrElse فيقوم بالتأكد من الشرط الأول فإذا كان صحيحاً فيقوم بإظهار النتيجة صحيحة بدون التأكد من الشرط الثاني.

استخدام Select Case

باستخدام الأداة الشرطية Select Case والتي تعني "اختر الحالة"، نعطي للكود البرمجي سهولة أكثر في القراءة والمراجعة، استخدمنا الأداة الشرطية Select Case في الفصلين الثالث والخامس من هذا الكتاب، وعندما يكون لدينا قائمة List Box أو Combo Box وتفيدنا كثيراً إذا كان لدينا متغير واحد وبخيارات متعددة. الهيئة العامة لـــ Select Case كالتالي:

Select Case Variable

Case Values1

Statment1

Case Value2

Statment2

Case Value3

Statment3

End Select

لاحظ بان Select Case تبدأ بـــ Select Case وتنتهي بـــ End Select وكلمة Variable نستعيض عنها بـــ متغير معين أما كلمة Value1 فمعناها قيمة المتغير في الحالة الأولى، وكذلك Value2 قسمة المتغير في الحالة الثانية. فإذا طابقت قيمة المتغير في الحالة الأُولى فإن البرنامج سينفذ الأمر الموجود بدل الكلمة Statment1 وإذا طابقت قيمة المتغير في الحالة الثانية Value2 فإن البرنامج سينفذ الجملة البرمجية الموجودة بدل Statment2 وهكذا.

فلنأخذ هذا المثال وهو عبارة عن ترجمة لما تعلمناه عن الأداة الشرطية Select Case:

لنفرض أن المستخدم سيقوم بإدخال العمر (عمر المستخدم)، فإذا أدخل المستخدم العمر 12 نريد أن تظهر له رسالة تقول له "أنت قاصر بحسب القانون"، أما إذا أدخل العمر 15 فتظهر له رسالة تقول له "أنت مكلَّف بحسب القانون المدني"، وإذا ادخل العمر 18 تظهر له رسالة تقول له "الآن يحق لك امتلاك وإدارة الشركات"، وإذا أدخل العمر 40 تظهر له رسالة تقول له "دخلت مرحلة الوقار" وإذا أدخل العمر 65 تظهر الرسالة "بلغت عمر التقاعد". لنقوم بتطبيق المثال أعلاه:-

```
Dim Age As Integer
Age = 18
Select Age
   Case 12
      MsgBox("أنت قاصر بحسب القانون")
   Case 15
      MsgBox("أنت مكلَّف بحسب القانون المدني")
   Case 18
      MsgBox("الآن يحق لك امتلاك وإدارة الشركات")
   Case 40
      MsgBox("دخلت مرحلة الوقار")
   Case 65
```

```
        MsgBox("بلغت عمر التقاعد")
    End Select
```

لاحظ بأننا حددنا العمر بـــ 18 في بداية الكود لكنه في نفس الوقت بإمكاننا جعل العمر في TextBox قابل للتغيير من قبل المستخدم وسيقوم البرنامج بإظهار رسائل الحوار تبعاً للأرقام التي سوف يكتبها المستخدم.

كما تدعم أداة الشرط Select Case استخدام Case Else قبل كتابة End Select، فبإمكاننا كتابة الجملة البرمجية كاملة بـــ Select Case وإذا لم تطابق أي من المدخلات من قبل المستخدم فإنه سيعمل بالأمر البرمجي الموجود بعد Case Else ومعناها (أي حالة أُخرى غير مذكورة). وبإمكاننا كذلك استخدام المعاملات المنطقية (=، <>، >، <، >=، <=) في الجمل البرمجية الشرطية بـــ Select Case ولكن بعد إضافة Is بعد كلمة Case. نطبق بالمثال التالي:

```
    Dim Age As Integer
    Age = 66
    Select Case Age
        Case Is < 15
            MsgBox("أنت قاصر بحسب القانون")
        Case 15
            MsgBox("أنت مكلَّف بحسب القانون المدني")
        Case 18
            MsgBox("الآن يحق لك امتلاك وإدارة الشركات")
        Case 40
            MsgBox("دخلت مرحلة الوقار")
        Case Is >= 65
```

```
MsgBox("بلغت عمر التقاعد")
Case Else
MsgBox("تأكد من الرقم المدخل، هناك خطأ ما")
End Select
```

نستخدم Select Case إذا كان لدينا متغير واحد وله ثلاث أو أكثر من ثلاث حالات أو قيم، أما إذا كان لدينا متغير وله حالتين أول أقل فيفضل استخدام If..Then .

مثال:

أنشئ مشروعاً جديداً وسمه My Select Case ، قم بإضافة أربعة ليبلات وListBox ، و زر Button إلى الفورم ثم قم بتعديل الخصائص لتتناسب مع الشكل أدناه (من دروسنا السابقة أعتقد أنه لديك القدرة على التعامل مع الخصائص، إذا أحسست أن هناك شيء صعب ارجع من جديد من أول الكتاب)



نضغط Double-Click على الفورم فيحولنا إلى صفحة الكود وبالتحديد إلى الحدث Load التابع للفورم، نكتب فيه الكود التالي، لتحميل العناصر في قائمة ListBox:

```
ListBox1.Items.Add("الدول العربية")
```

```
ListBox1.Items.Add("England")
ListBox1.Items.Add("Germany")
ListBox1.Items.Add("Mexico")
ListBox1.Items.Add("Italy")
```

ثم نضغط على الزر ونكتب الكود التالي لإغلاق البرنامج:

```
End
```

وكذلك نضغط على القائمة ListBox لنتحول مباشرة إلى منطقة الكود وبالضبط إلى الحدث SelectedIndexChanged التابع للقائمة ListBox1 ونكتب فيها هذا الكود الذي يحدد ماذا اختار المستخدم من بنود القائمة ويكتب جملة الترحيب في Label4 وكذلك يكتب نص الاختيار في Label3:

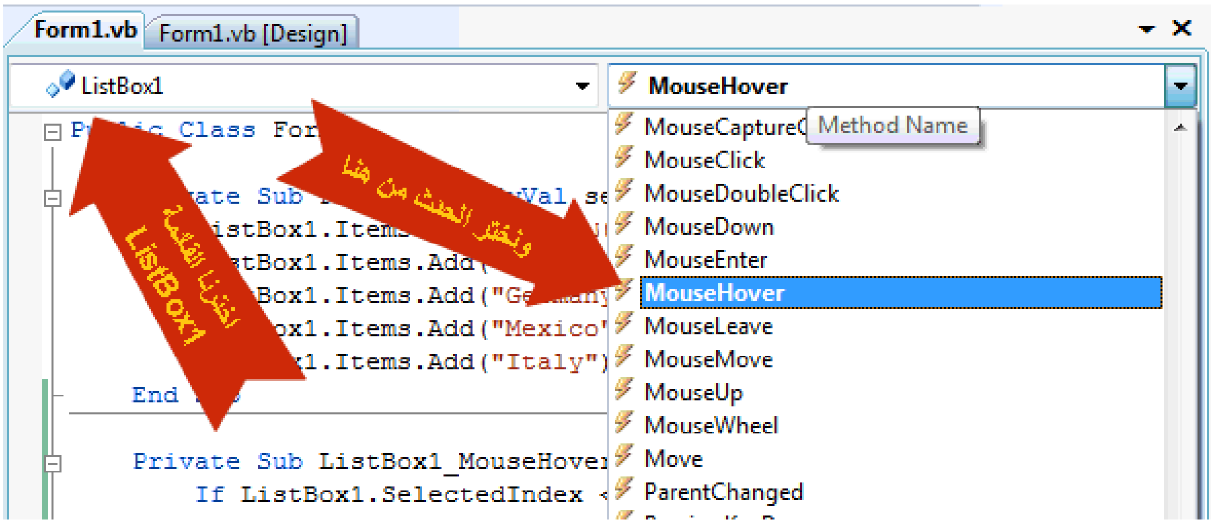
```
Select Case ListBox1.SelectedIndex
    Case 0
        Label3.Text = ListBox1.SelectedItem
        Label4.Text = "أهلا، عزيزي المبرمج"
    Case 1
        Label3.Text = ListBox1.SelectedItem
        Label4.Text = "Hello, programmer"
    Case 2
        Label3.Text = ListBox1.SelectedItem
        Label4.Text = "Hallo, programmierer"
    Case 3
        Label3.Text = ListBox1.SelectedItem
        Label4.Text = "Hola, programador"
```

```
Case 4
    Label3.Text = ListBox1.SelectedItem
    Label4.Text = "Ciao, programmatore"
End Select
```

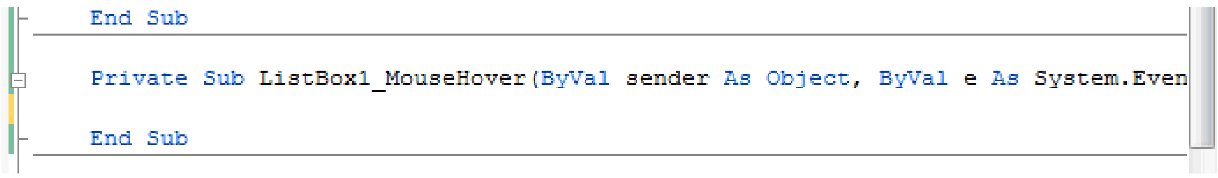
قم بتشغيل البرنامج وجرب أختر أية بند من بنود القائمة ListBox ماذا تلاحظ، سنلاحظ انتقال اسم البند إلى Label3 وكذلك ظهور جملة في Label4 بحسب الجُمل التي أضفناها في منطقة الكود. ستجد نسخة من المشروع أعلاه مع المرفقات برقم 018. الآن انتهينا من الجمل البرمجية الشرطية باستخدام If..Then وكذلك باستخدام Select Case إذا كان لديك صعوبة عُد إلى الشرح ثانية لفهم طبيعة If..Then و Select Case لأننا بالطبع سنتحاج لها كثيراً في هذا الكتاب، ومع تقدمك في عالم البرمجة ستجد إن If..Then و Select Case لا يمكن أن يُستغنى عنها في أي برنامج من البرامج.

التعرف على تحرك الماوس – أحداث الماوس Hovers

بدأنا في هذا الفصل في التعرف على الأحداث التابعة للمكونات البرمجية المتوفرة في بيئة التطوير والتي يتعامل معها الفيجوال بيسك، ومع تقدمنا في هذا الفصل عرفنا كيف يتم التعامل مع الأحداث بالربط الشرطي باستخدام If..Then و Select Case، أما الآن سنتعرف كيف نتعامل مع أحداث الماوس والتي سوف نراقب حركة الماوس Mouse Hover فوق قائمة الدول المتوفرة في القائمة ListBox والموجودة ضمن المثال رقم 018، نفتح المثال المذكور ونذهب إلى محرر الكود ونختار المكون ListBox1 لنذهب إلى الحدث Mouse Hover كما في الصورة التالية:



وبعد الاختيارات المحددة في الصورة أعلاه تقوم بيئة التطوير بكتابة الكود في الصورة في التالية في منطقة الكود:



نضيف الكود التالي بين Private Sub و End Sub:

```
If ListBox1.SelectedIndex < 0 Or ListBox1.SelectedIndex > 4 Then
        Label3.Text = "قم باختيار بند من القائمة"
End If
```

فمع العلم بأن البنود المختارة من القائمة ListBox1 تتراوح أرقامها (أرقام ترتيبها في القائمة) بين 0 و 4 فتم استخدام الجملة الشرطية If..Then لمعرفة إذا ما لم يتم اختيار بند من الــ 0 إلى 4 فتظهر لنا رسالة في Label3 تقول "قم باختيار بند من القائمة"، لنقوم على أساسها باختيار بند من القائمة، كيف تظهر هذه الرسالة! من الكود أعلاه قمنا بمراقبة الحدث Mouse Hover لمعرفة حركة الماوس فوق القائمة فإذا تحرك الماوس فوق القائمة ولم يكن هناك بند قد تم اختياره من بنود القائمة (البنود من 0 إلى 4) ستظهر تلك الرسالة في Label3. ستجد نسخة من المشروع المعدل في المرفقات برقم 018b.

خلاصة الفصل السادس:

من اجل أن	**قم بالتالي**
لكتابة جملة شرطية	استخدم احد المعاملات =، <>، >، <، >=، <= للربط بين قيمتين.
لاستخدام If..Then	If *condition1* Then Statements1 ElseIf *condition2* Then Statements2 Else Statements3 End If حيث سيتم تنفيذ الأمر في Statements1 إذا طابقت الحالة مع *condition1* وسيتم تطبيق الأمر في Statements2 إذا طابقت الحالة مع *condition2* وإذا لم تطابق الحالتان الأوليتان سيتم تنفيذ الأمر في Statment3.
استقبال المدخلات من المستخدم على هيئة أو صيغة محددة.	نستخدم MaskedTextBox ونضبط خصائصه على الهيئة التي نريد.
لاستخدام Select Case	Select Case *variable* Case value1 Statement1

	Case value2 Statement2 Case Else Statement3 End Select حيث نكتب المتغير بدلاً من variable فإذا طابقت القيمة 1 Value1 مع المتغير سيتم تطبيق الأمر في الجملة Statement1 وإذا طابقت القيمة 2 Value2 سيتم تطبيق الأمر في الجملة Statement2 وإذا لم تطابق القيمتان أعلاه سيتم تنفيذ الأمر في الحملة Statement3.
لدمج جملتين شرطيتين	نستخدم المعاملات المنطقية للدمج And, Or, Not, Xor
لزيادة سرعة تنفيذ البرنامج عند مقارنة الجمل الشرطية	نستخدم الأدوات الشرطية AndAlso و OrElse لأن البرنامج سيقوم بالتأكد من الشرط الأول فإذا لم يطابق في حالة AndAlso فإنه مباشرة وقبل مطابقة الشرط الثاني سيظهر نتيجة عدم المطابقة أو False، أما في حالة OrElse فإن البرنامج سيقوم بمطابقة الشرط الأول وفي حالة المطابقة فإنه ومباشرة وبدون التحقق من الشرط الثاني سيطهر نتيجة المطابقة الإيجابية أو True.
كتابة كود لحدث ما	نذهب إلى محرر الكود ومن أعلى يمين منطقة الكود نختار المكون المراد تتبع أحداثه ثم نذهب إلى اليمين ونختار الحدث المراد وسيقوم الفيجوال بيسك مباشرة بكتابة كود الحدث، ما علينا إلا أن نكتب الكود المراد تنفيذه بين Private Sub وبين End Sub.

الفصل السابع

الحلقات التكرارية والمؤقتات Loops and Timers

يعد استخدام الحلقات التكرارية For…Next هاماً جداً بسبب توفير العديد من الكود فبدلاً من تكرار الكود عدة مرات نستخدم حلقة تكرارية For…Next لاختصار الكود. الشكل العام لجملة For…Next البرمجية كالتالي:

```
For Variable = start to end
Statment1
Statment2
Statment3
Next [Variable]
```

حيث يتم أولاً تعريف المتغير Variable ومن ثم استخدامه في حلقة التكرار، ونكتب بداية الحلقة التكرارية بدلاً من start ونهايتها بدلاً من end ثم نكتب الأوامر البرمجية بدلاً من Stament1، Statment2، Statment3 وسيتم تنفيذها من قبل البرنامج الواحدة تلو الأخرى. لنأخذ مثال بسيط على الحلقات التكرارية:

```
Dim i As Integer
For i = 1 To 4
   Beep()
Next i
```

في الكود أعلاه غرفنا المتغير i على إنه عدد صحيح ثم فتحنا حلقة تكرارية من 1 إلى 4 وسيتم تنفيذ الأمر Beep() عند كل مرة أي أن البرنامج سيقوم بتنفيذ الأمر Beep() أربع مرات بسبب وجود الحلقة التكرارية، فبدلاً من كتابة الأمر Beep() أربع مرات تم ربطه بحلقة تكرارية For…Next، تخيل إذا لم توجد الحلقات التكرارية في بيئة التطوير فسيكون علينا كتابة ألأمر أعلاه كما يلي:

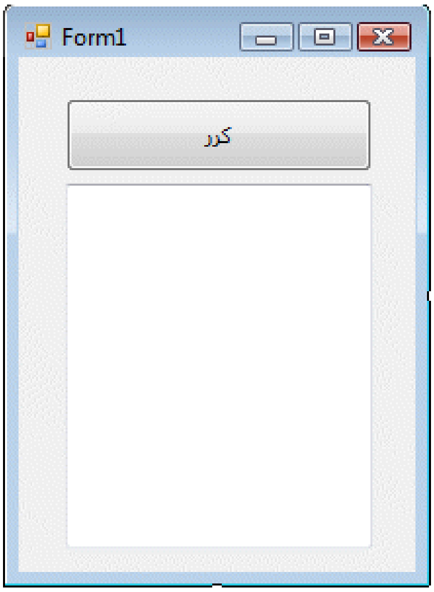
```
Beep()
Beep()
Beep()
```

Beep()

قد يقول قائل"لا داعي للحلقات التكرارية سأقوم بكتابة المر البرمجي أربع، ثلاث أو خمس مرات" نقول له ماذا ستفعل إذا كان لابد عليك من تكرار عملية ما مائة مرة أو أكثر في مثل هذه الحالة ستكون مضطراً لاستخدام الحلقات التكراريةFor...Next. تستطيع استفادة الاستفادة القصوى من الحلقات التكرارية وسنوفر الكثير من الجهد. ستلاحظ هذا في ما يلي من الشرح عن الحلقات التكرارية.

مثال على استخدام الحلقات التكرارية لكتابة أكثر من سطر نصي في مربع النص

أولاً علينا أن نعرف أننا نستطيع أن نسند مخرجات الحلقات التكرارية إلى أي خاصية من الخصائص أو إلى صندوق نص معين أو إلى مخرجات على هيئة معينة أو إلى مدخلات في برنامج أخر قد يكون متصفح إنترنت أو أي قاعدة بيانات، فكرة كتابة أكثر من سطر نصي في صندوق النص تتم بواسطة تغيير الخاصية Multiline لصندوق النص من False إلى True فقط وبكل سهولة. أنشئ الآن مشروعاً جديداً وليكن اسمه For Loop بعد ظهور الفورم الفارغ في بيئة التطوير قم بإضافة زر Button إلى الفورم لكن هذه المرة ستقوم بذلك بطريقة جديدة ليست طريقة السحب والإلقاء التي تعودت عليها ولكن بواسطة الضغط Double-Click على الزر Button في نافذة الأدوات فسيقوم الفيجوال بيسك مباشرة بإضافة الزر إلى أعلى يسار الفورم، فطريقة الـــ Double-Click تزيد من إنتاجيتنا وقت تصميم البرنامج. نضغط Double-Click كذلك على مربع النص TextBox في صندوق الأدوات ليقوم الفيجوال بيسك بإضافة صندوق نص إلى الفورم نغير الخاصية Multiline لصندوق النص من False إلى True ونقوم بتعديل خاصية الـــ Text للزر Button إلى "كرر" بعد التعديلات أعلاه وبعد تغيير أماكن الأدوات على الفورم سيكون الفورم بهذا الشكل:



نضغط Double-Click على الزر "كرر" لنكتب هذا الكود في محرر الكود:

```
Dim i As Integer
Dim Wrap As String
Wrap = Chr(13) & Chr(10)
For i = 1 To 10
   TextBox1.Text = TextBox1.Text & "Line " & i & Wrap
Next i
```

في السطر الأول من الكود عرفنا i على أنه عدد صحيح وفي السطر الثاني عرفنا Wrap على انه متغير من النوع String (وللعلم النوع String متغير خاص بالنصوص ويعني "نص" أي نص حتى الضغط على Enter في الكيبورد يعتبر نص المسافة Space في الكيبورد تعتبر نص الخ). حددنا قيمة Wrap بأنه يساوي (Chr(13 و (Chr(10 في الكيبورد فــ (Chr(13 تنقل مؤشر الكتابة إلي بداية السطر أما (Chr(10 فتنقل بالكتابة من سطر إلى آخر في صندوق النص. عند تنفيذ البرنامج يقوم البرنامج بكتابة Line عشر مرات في صندوق النص لكن بالاستعانة بالمتغير Wrap يتم الانتقال من سطر إلى آخر في صندوق النص، وبسبب المتغير i يتم كتابة رقم مع Line في كل سطر يعكس هذا الرقم رقم i في السطر المحدد ولأن i تساوي من 1 إلى 10

فكل سطر من الأسطر سيحتوي على رقم من الأرقام من 1 إلى 10 بجانب كلمة Line لأن & تقم بعملية الربط بين النصوص بداخل مربع النص، إذا رأيت أن هذا عسير على الفهم قم بقراءته ثانية ثم ادرس الكود من جديد لمعرفة كيف تم ذلك. ستجد نسخة من التطبيق أعلاه في المرفق رقم 019. بعد تشغيل البرنامج يمكنك الضغط على زر كرر أكثر من مرة وفي كل مرة سيقوم البرنامج بإضافة عشرة أسطر جديدة إلى صندوق النص ولن نرى التغيير بسهولة بسبب تكرر نفس الكلمة ولعدم وجود أشرطة التمرير في صندوق النص ولإضافة أشرطة التمرير إلى صندوق النص نذهب إلى خصائص صندوق النص ونختار الخاصية Scrollbars ونغيرها من None إلى Vertical ليقوم الفيجوال بيسك بإضافة أشرطة تمرير عمودية لصندوق النص. للعلم صندوق النص TextBox لا يسمح بكتابة نصوص كثيرة فيه وأقصى حد ممكن للنص بداخله ما يساوي 65 kb والتي تساوي خمسي ألف حرف أو أكثر تقريباً، وفي حالة أردنا كتابة نصوص أكثر في صندوق النص من هذا الحد فما علينا إلا أن نستخدم أداة صندوق النص RichTextBox التي تستوعب الكثير من النص مع العديد من الخيارات (خيارات النص).

تصميم حلقات تكرارية For...Loop احترافية

بسبب تعدد استخدامات الحلقات التكرارية For...loop، هناك العديد من المناسبات التي نحتاج لتعديل الحلقة التكرارية لتنفيذ التكرار ولكن بشروط معينة، فمثلاً يمكننا تغيير نقطة البداية وكذلك نقطة النهاية وفي نفس الوقت يمكننا تغيير مقدار الخطوة أو القفزة من رقم إلى آخر دعونا نأخذ هذا الكود ونضيفه للتطبيق السابق بدلاً عن الكود السابق:

```
Dim i As Integer
Dim Wrap As String
Wrap = Chr(13) & Chr(10)
For i = 5 To 25 Step 5
   TextBox1.Text = TextBox1.Text & "Line " & i & Wrap
Next i
```

بعد تنفيذ البرنامج والضغط على الزر "كرر" سيقوم البرنامج بكتابة التالي:

Line 5

Line 10

Line 15

Line 20

Line 25

لماذا! لأننا حددنا نقطة بداية التكرار 5 ونقطة الانتهاء 25 ومقدار الخطوة بين كل تكرار والذي يليه 5، نستطيع كذلك جعل الخطوة عبارة عن رقم عشري فنكتب الكود أعلاه :

```
Dim i As Single
Dim Wrap As String
Wrap = Chr(13) & Chr(10)
For i = 1 To 2.5 Step 0.5
    TextBox1.Text = TextBox1.Text & "Line " & i & Wrap
Next i
```

فستكون المخرجات في صندوق النص كالتالي:

Line 1

Line 1.5

Line 2

Line 2.5

حلقات التكرار نستطيع استخدامها في العديد من المهام التي نحتاجها في برامجنا، فالمسألة ليست تكرار أرقام وفقط لكن يمكننا استخدامها في تكرار عرض صور معينة على الشاشة أو غيرها من الاستخدامات لنأخذ مثلاً هذا التطبيق الذي يمكننا من عرض العديد من الصور على الفورم واحدة تلو الأخرى يفصل بينهم صندوق حوار Message Box وبعد أن يقوم المستخدم بالضغط على صندوق الحوار يقوم البرنامج بإظهار الصورة التالية. لننشئ تطبيقاً جديداً باسم For...Loop Icons ونضيف زر Button للفورم ونضبط الخاصية Text على "أظهر الأربعة الوجوه"

وصندوق للصورة Picture Box ونضبط الخاصية SizeMode على StretchImage ثم نضغط Double-Click على الزر ونكتب الكود التالي في محرر الكود:

```
Dim i As Integer
For i = 1 To 4
   PictureBox1.Image = System.Drawing.Image.FromFile _
 ("C:\Users\Marwan\Desktop\vbsbs\020\face\face0" & i & ".ico")
   MsgBox("اضغط هنا لعرض الوجه التالي")
Next
```

لاحظ هنا أن السطر الرابع من الكود معني بتحديد مسار وجود الإيقونات التي تبدأ أسمائها بـــ face01 إلى face04 (بالطبع سيختلف مسار الأيقونات في جهازك عن المسار أعلاه فتنبه لذلك). قم بتنفيذ البرنامج ولاحظ طيف يقوم البرنامج بعرض الأيقونات واحدة تلو الأخرى في صندوق الصورة.

ملاحظة

في الكود أعلاه السطرين الثالث والرابع عبارة عن سطر واحد ولكن بسبب طول السطر في محرر الكود قسمناه إلى سطرين (مع إمكانية عدم تقسيمه إلى سطرين) بواسطة طباعة مسافة في سطر الكود ثم الإشارة _ ثم Enter بعدها مباشرة. نستطيع تقسيم أي سطر برمجي طويل بنفس هذه الطريقة.

ملاحظة

قمنا بعرض صندوق حوار بين كل صورة وأُخرى لإبطاء عرض الصور، لان طريقة عرض الصور ستكون سريعة جداً لدرجة غير مرئية.

توجد نسخة من التطبيق أعلاه في المرفق رقم 020.

نستطيع جعل المستخدم يقوم بتغيير الصورة ليس عن طريق الموافقة على صندوق النص Message Box ولكن بواسطة الضغط مرة أُخرى على الزر Button في هذه الحالة نقوم بتعريف المتغير i كالتالي:

```
Dim i As Integer = 1
```

ولكن لابد أن نقوم بتعريفه كمتغير عام على طول الفورم، بعبارة أُخرى لابد أن نقوم بتعريفه تحت الـــ Public Class Form1 مباشرة، بعد كتابة التعريف بالمتغير والضغط على Enter نلاحظ أن بيئة التطوير قد أضافت سطر للتفريق بينه وبين بقية الدوال والأوامر البرمجية في محرر الكود وتسمى منطقة التعريف تلك منطقة "التعريفات". تحت الحدث Click التابع للزر Button1 نحذف الكود الموجود ونكتب بدلاً عنه هذا الكود :

```
PictureBox1.Image = System.Drawing.Image.FromFile _
  ("C:\Users\Marwan\Desktop\vbsbs\020\face\face0" & i & ".ico")
     i = i + 1
     If i = 5 Then
       i = 1
    End If
```

الملاحظ في الكود أعلاه أننا نزيد قيمة المتغير i رقم واحد مع كل عملية ضغط على الزر وعلية فسيقوم التطبيق بإظهار صورة جديدة كل مرة وعند وصول البرنامج إلى الصورة الرابعة سيعود من الصورة الأولى لأننا كتبنا الشرط If إذا كانت i تساوي 5 فقم بتغيير القيمة إلى 1، وعلى هذا يقوم البرنامج بعرض الصور من 1 إلى 4 بشكل متسلسل. ملاحظة هنا بأننا نستطيع كتابة الكود:

```
i = i + 1
```

بهذا الشكل:

```
i +=1
```

لإن i = i + 1 هي نفسها i +=1

التطبيق أعلاه هو عبارة عن التطبيق رقم 020 ويوجد التعديل بالمرفقات بالرقم 020b

إذا ظهرت لك رسالة خطأ وقت تنفيذ البرنامج فتأكد من مسار الأيقونات في جهازك أنه المسار الصحيح المذكور في الكود وإلا قم بتغيير الكود ليتناسب مع المسار الجديد للملف.

إنهاء الحلقات التكرارية بدون تكملة التكرار

إذا أردنا إنهاء الحلقات التكرارية عند تحقق شرط معين علينا تحديد ذلك الشرك في الجمل التكرارية For...Next ثم كتابة Exit For وسيقوم البرنامج مباشرة بالتوقف عن التكرار والانتقال أو تنفيذ السطر البرمجي الذي يلي الحلقة التكرارية. لنأخذ هذا المثال الذي يظهر لنا صندوق مدخلات InputBox عشر مرات وفي حالة قام المستخدم بإدخال الكلمة "انتهى" سيتوقف عن التكرار (قبل ظهور العشر تكرارات) حتى إذا كانت المرة الثانية أو الثالثة من التكرار:

```
Dim i As Integer
Dim InpName As String
For i = 1 To 10
    InpName = InputBox("Enter your name or type Done to quit.")
    If InpName = "انتهى" Then Exit For
    TextBox1.Text = InpName
Next i
```

إذا قام المستخدم بإدخال الكلمة "انتهى" في صندوق المدخلات InputBox سيقوم البرنامج بترك التكرار والانتقال إلى السطر البرمجي الذي يليه (أي إلى السطر الذي يلي Next i).

الحلقات التكرارية بطريقة أُخرى Do...Loop

قد نحتاج لتكرار أمر ما إلى حين معين أو إلى تنفُذ شرط معين، في هذه الحالة نستخدم Do...Loop وهي عبارة عن بديل للتعبير For...Next، فيتم تنفيذ تكرار معين حتى يتحقق شرط ما ثم يتوقف التكرار. Do...Loop هامة جداً وضرورية لأننا في بعض الأحيان قد لا نعرف كم

عدد مرات التكرار التي نريدها، فإذا كنا في تطبيق ما نريد أن نستخدم تكرار ما لمرات غير معلومة (ممكن لمرتين أو لثلاث مراث أو خمسة وعشرون مرة) ونريد أن يتوقف التكرار فقط عند كتابة المستخدم الكلمة "انتهى" في صندوق الإدخال InputBox لنأخذ هذا المثال:

```
Dim InpName As String
Do While InpName <> "انتهى"
    InpName = InputBox("اكتب الأسم أو اكتب الكلمة ""انتهى"" للخروج.")
    If InpName <> "انتهى" Then TextBox1.Text = TextBox1.Text &
Chr(13) & Chr(10) & InpName
Loop
```

فسيقوم البرنامج بإضافة أي اسم تتم كتابته إلى صندوق النص TextBox1 حتى يتم كتابة الكلمة "انتهى" في صندوق النص في حينه فقط سيغلق صندوق النص. لاحظ في الكود أعلاه أننا كتبنا الكلمة "انتهى" داخل علامتين من علامات التنصيص كالتالي ""انتهى"" لإن بيئة التطوير لا تقوم بإضافة علامات التنصيص إلى رسائل الخوار مع المستخدم إلا إذا كررناها مرتين. تنبه هنا أن الكلمة "انتهى" حساسة لحالة الأحرف إذا كانت بالإنجليزية وكذلك حساسة لنوع الأحرف باللغة العربية، فلا يمكننا كتابة أنتهى أو انتهي بدلاً عن "انتهى" وكذلك لا يمكننا كتابة Done أو DONE بدلاً عن done (إذا كانت الكلمة done بدلاً عن انتهى)، بالنسبة للغة الإنجليزية نستطيع تعديل خاصية حالة الأحرف بواسطة الدالة StrComp التي سوف نتناولها في الفصل الثالث عشر من هذا الكتاب بعون الله، أما بالنسبة للغة العربية فلا بد من كتابة كل كلمة نريدها في منطقة الكود كالتالي:

```
If InpName <> "انتهى" Or “انتهي” Then TextBox1.Text = TextBox1.Text &
Chr(13) & Chr(10) & InpName
```

فقط أضفنا الكلمة "انتهي" لتوقعنا بأن بعض المستخدمين سوف يستخدمها لإنهاء التكرار Do...Loop وكذلك يمكننا كتابة جميع الكلمات المتوقعة والمقاربة لـــ انتهى في منطقة الكود وربط كل الكلمات بالمعامل المنطقي Or أو تصميم دالة للتعامل مع الكلمات والأحرف المتشابهة.

تجنب التكرارات الغير منتهية

عند تصميم تكرار معين لابد لهذا التكرار أن تكون له نقطة نهاية لإن التكرارات التي لا نهاية لها قد تجعل من برنامجنا عبئاً كبيراً على الذاكرة المؤقتة (الرام) وقد تجعل برنامجنا لا يستجيب للمستخدم. تخيل معي هذا التكرار:

```
Dim Number as Double
Do
    Number = InputBox("اكتب أي رقم لتتم كتابة مربعة. اكتب 1- للخروج.")
    Number = Number * Number
    TextBox1.Text = Number
Loop While Number >= 0
```

لاحظ الكود أعلاه أننا في الرسالة كتبنا للمستخدم اكتب "–1" للخروج لكننا لم نترجم ذلك فعلاً في منطقة الكود لإن مربع الـــ "–1" يساوي 1 وعلية سيظل برنامجنا بإيجاد المربع لأي رقم سوف يدخله المستخدم حتى قيام الساعة. لكن بإمكاننا تحديد طريقة أُخرى لإغلاق التكرار.

> **ملاحظة:**
>
> تأكد من وجود نهاية منطقية للتكرارات الموجودة في برنامجك.

استخدام Loop...Until

إذا كان لدينا تكرار معين ونريده أن يتوقف عند تحقق شرط معين نستخدم Loop...Until كما في المثال التالي والذي سيتوقف التكرار عند كتابة المستخدم الكلمة انتهى في صندوق المدخلات InputBox :

```
Dim InpName As String
    Do
 InpName = InputBox("اكتب أي كلمة أو اكتب انتهى لإغلاق التكرار")
```

```
If InpName <> "Done" Then TextBox1.Text = InpName
Loop Until InpName = "انتهى"
```

من الملاحظ أن Loop...Until هي نفسها Loop...While مع فرق بسيط فقط في المعامل في جملة الشرط في السطر الأخير فبدلاً من كتابة = مع Loop...Until نكتب <> مع Loop...While. وتستطيع استخدام أي منهما مع التنبه لمعامل الشرط المذكور.

الأداة Timer المؤقت:

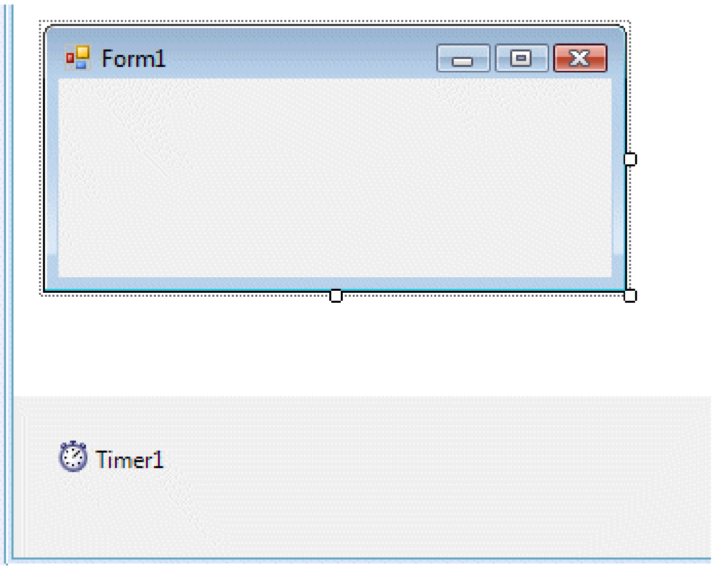
المؤقت أداة عظيمة يمكننا الاعتماد عليها في الكثير من المهمات البرمجية، فالمؤقت هو عبارة عن ساعة إيقاف مخفية تستطيع التعرف على ساعة النظام والاستفادة من ذلك في برامجك. تستطيع الاستفادة من أداة المؤقت في الكثير من الأمور بتكرار أمر برمجي معين بين كل وقت وآخر أو بحساب ساعات العمل على برنامج معين أو بحساب الوقت منذ تشغيل البرنامج أو بمعرفة سلوكيات المستخدم المرتبطة بالوقت وغيرها العديد من المهام البرمجية التي لها علاقة بالوقت.

وللمؤقت Timer خاصيتين مهمتين وهما الخاصية Interval وهي التي تحدد خطوة المؤقت بالملي ثانية والخاصية الثانية هي Enabled وهي التي تفعل أو توقف المؤقت.

تصميم ساعة في برنامجنا باستخدام الأداة Timer

بإمكاننا تصميم ساعة رقمية في برنامجنا بالاستفادة من الأداة Timer أو أداة المؤقت كالتالي:

ننشئ مشروعا جديداً ونسميه Digital Clock ، ثم نصغر الفورم إلى مستطيل صغير ونضيف له المؤقت Timer من صندوق الأدوات ولإن المؤقت يعتبر من المكونات Components (المكونات Components هي عبارة عن أدوات في بيئة التطوير ولكنها تعمل من خلف الكواليس بدون أن تكون موجودة على الفورم) فلن يظهر فوق الفورم ولكن سيظهر تحت الفورم كما في الصورة:



هل تتذكر الفصل الرابع عندما قمنا بإضافة قائمة Menu لقد قامت بيئة التطوير بإضافتها إلى نفس المكان التي أضافت اليوم فيه المؤقت. قم بإضافة ليبل Label وغير الخاصية AutoSize من True إلى False ثم قم بتكبير الليبل ليملأ المكان الفارغ في الفورم، وغير كذلك الخاصية Text واجعلها فارغة، ثم غير خاصية Text في الفورم إلى "الساعة الرقمية" أما المؤقت فقم بتغيير الخاصيتين Enabled إلى True والخاصية Interval إلى 1000.

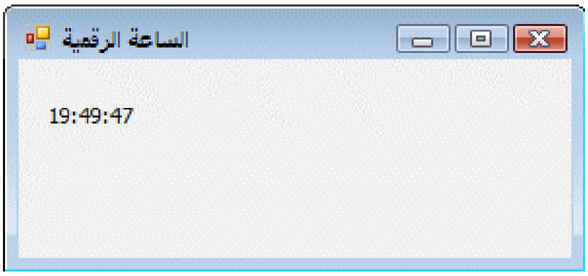
ملاحظة:

إذا أردت تغيير خلفية الفورم، فقط اذهب إلى الخاصية **BackgroundImage** التابعة للفورم ثم اختر منها صورة محددة في جهازك.

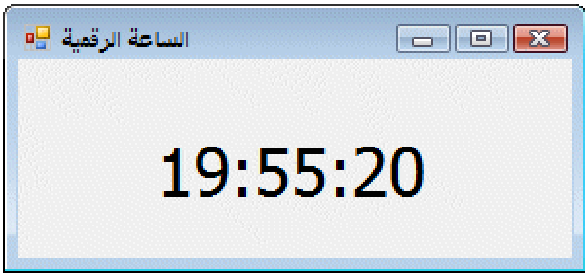
–الآن ننتقل إلى مرحلة الكود، اضغط Double-Click على المؤقت Timer1 ستنتقل مباشرة إلى محرر الكود وبالتحديد إلى الحدث Tick التابع للمكون Timer1، هنا سنكتب الكود الذي نريد من المؤقت تنفيذه عند كل 1000 ملي ثانية وهو كالتالي:

```
Label1.Text = TimeString
```

الكود أعلاه يقوم بكتابة الوقت بطريقة (الساعة والدقيقة والثانية) في نص الموجود في الليبل Label1. عند تنفيذ البرنامج سنلاحظ أن الليبل يحتوي على الساعة ويتم تغييرها كل 1000 ملي ثانية (أي عند كل ثانية). لاحظ الصورة:



لكن تتم كتابة الوقت بخط صغير، كيف نقوم بتكبير النص في داخل Label1، ما علينا إلا أن نذهب إلى خصائص Label1 ثم نغير الخاصية Font إلى Tahoma, 26.25pt لتكبير الخط وكذلك نغير الخاصية TextAlign إلى MiddleCenter لتوسيط الكتابة في وسط الليبل. بعد التعديلات سيظهر الوقت كالصورة التالية:



طبعاً إذا أردنا إظهار التاريخ بدلاً من الوقت فيمكننا كتابة Today بدلاً من TimeString ، جرب استخدام TimeOfDay أو DateString ، ماذا ستلاحظ! وما الفرق بين كل كلمة وأخرى. ستجد نسخة من التطبيق أعلاه في المرفق رقم 021.

استخدام المؤقت لتحديد فترة زمنية

كما وضحنا سابقاً بأننا نستطيع استخدام المؤقت لكل ما له علاقة بالوقت أو التوقيت، فيمكنك حفظ ما يقوم المستخدم بطباعته كل عشر دقائق أو تحديد مدة زمنية لإدخال كلمة السر أو إظهار رسالة ترحيبية لمدة معلومة عند تشغيل البرنامج الخ. لنأخذ الآن مثال عن برنامج يقوم بطلب كلمة سر ولابد على المستخدم أن يقوم بإدخال كلمة السر الصحيحة خلال 15 ثانية (طبعاً كلمة السر الصحيحة ستكون 123456)، وإذا تأخر المستخدم عن إدخال كلمة السر خلال 15 ثانية فسيغلق البرنامج (بالطبع سنحتاج مثل هذه الفكرة عند تصميم البرامج الكبيرة). لنأخذ المثال:

١-نقوم بإنشاء تطبيق جديد ونسميه Timed Password

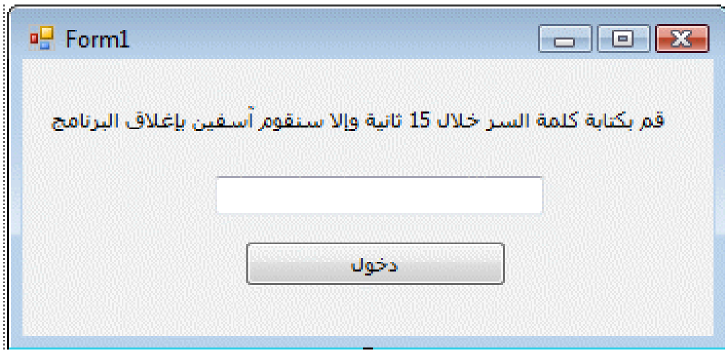
٢-قم بتغيير الفورم لنفس حجم الفورم الذي أخذناه في المثال السابق، ثم قم بإضافة ليبل Label وصندوق للنص TextBox.

٣-أضف مؤقت إلى برنامجك واضبط خاصية الـ Interval على 15000 وخاصية Enabled على True

٤-كذلك قم بإضافة زر Button واضبط خاصية Text على "دخول"

٥-نقوم بتغيير الخاصية AcceptButton التابعة للفورم Form1 إلى Button1.

٦-نقوم بتغيير خاصية الـ Text لليبل على "إلى قم بكتابة كلمة السر خلال 15 ثانية وإلا سنقوم آسفين بإغلاق البرنامج"، وكذلك خاصية Password Char التابعة للمكون TextBox1 على "*". سيكون الفورم كما في الشكل التالي:



ملاحظة حول تغيير الخاصية Password Char التابعة لصندوق النص TextBox1 ليتلاءم صندوق النص مع كونه صندوق نصي لاستقبال كلمة السر، من أجل أن تكون كلمة السر عبارة عن نجوم. أما الخاصية AcceptButton التابعة للفورم فتعني ما هو الزر اللازم الضغط علية عند الضغط على Enter ونحن على الفورم. فنحن حددنا Button1 كزر افتراضي فعند ضغط المستخدم على Enter كأنه قام بالضغط على Button1 باستخدام الماوس. الآن نقوم بالضغط Double-Click على المؤقت ثم نكتب الكود التالي:

```
End
```

وهذا الكود يقوم بإغلاق البرنامج إذا تأخر المستخدم 15 ثانية. ثم نقوم بالضغط Double-Click على الزر Button1 ونكتب هذا الكود:

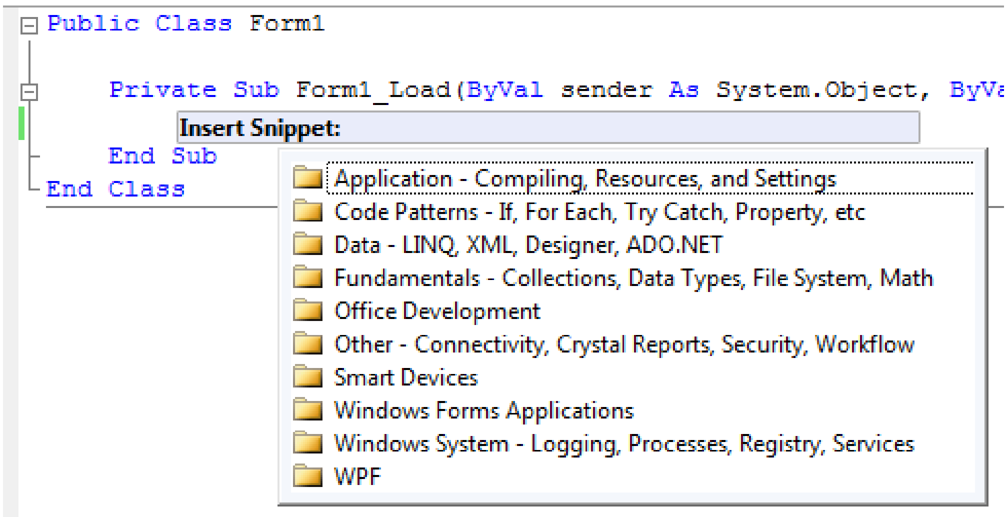
```
If TextBox1.Text = "123456" Then
   Timer1.Enabled = False
   MsgBox("شكراً على كتابة كلمة السر الصحيحة")
Else
   MsgBox("كلمة السر خاطئة، حاول مرة أُخرى")
   TextBox1.Text = ""
End If
```

ولشرح الكود أعلاه لابد أن نعلم أن كلمة المرور الصحيحة هي **123456** فإذا أدخل المستخدم هذه الكلمة فإن التطبيق سيقوم بإيقاف عمل المؤقت (العمل الذي سيقوم به بعد خمسة عشر ثانية وهو إيقاف البرنامج)، ثم سيقوم التطبيق بإظهار رسالة حوار فيها " شكراً على كتابة كلمة السر الصحيحة " أما إذا كانت كلمة المرور المدخلة غير فإن التطبيق سيقوم بإظهار رسالة حوار فيها "كلمة السر خاطئة، حاول مرة أُخرى"، ثم سيقوم بطمس الكلمة الخاطئة التي كتبناها. توجد نسخة من التطبيق أعلاه في المرفق رقم 022. عند تنفيذ البرنامج تذكر أنه لابد من إدخال كلمة السر الصحيحة (وهي 123456) خلال خمسة عشر ثانية فقط من تشغيل التطبيق.

من التعرف على المؤقت Timer تستطيع التخيل كم من المهام البرمجية التي قد تستخدم لها مؤقت، فيمكنك مثلاً لا حصراً استخدام المؤقت مع الجمل التكرارية التي تعرفنا على سابقاً في هذا الكتاب. أو استخدام المؤقت مع جملة **If** الشرطية الخ. بالربط بين المؤقت والخصائص والجمل التكرارية وكذلك المتغيرات تمتلك الآن قوة في عالم الفيجوال بيسك يمكنك تسخيرها لتصميم العديد من البرامج التي قد تحتاجها أو يحتاجها عملائك.

كيفية إضافة Snippets في مرحلة الكود:

الترجمة الحرفية لكلمة Snippets هي قصاصة لكن المقصود هنا بـــ Snippets هي الأكواد الموجودة أو الدوال المتوفرة مسبقاً ضمن بيئة التطوير. Snippets توفر لك الوقت فبدلاً من كتابة كود يقوم بإضافة زر Button أو TextBox وقت تنفيذ البرنامج وبدلاً من كتابة كود لنسخ أو قص نسخ بواسطة الكليبورد Clipboard، ما عليك إلا أن تضغط بالماوس Right-Click في منطقة الكود (طبعاً في المكان المناسب) وتختار من القائمة التي ستظهر Insert Snippet بعد ذلك ستظهر لك قائمة بالـــ Snippet المتوفرة كما في هذه الصورة:



تستطيع اختيار المجموعة المناسبة التي تريد منها الكود ثم الانتقال إلى مجموعات فرعية أصغر حتى تصل إلى الكود المطلوب. فمثلاً باختيار المجموعة قبل الأخيرة ستجد الكثير من الأكواد الخاصة برقم نسخة الويندوز وكذلك الخاصة باسم المستخدم والذاكرة المؤقتة (الرام) وغيرها من الأكواد الهامة التي قد تحتاجها.

لمعرفة الاختصارات Snippet عن قرب اذهب إلى القائمة Tools ثم اختر Code Snippets Manager أو اضغط Ctrl+K+B وسيظهر لك فورم تحتوي على كل الاختصارات Snippets مع شرح مبسط لكل كود متوفر، بالإضافة إلى إمكانية تغيير لغة البرمجة من فيجوال بيسك إلى سي شارب.

خلاصة الفصل السابع:

قم بالتالي	من اجل أن
اكتب الأمر البرمجي بين For و Next في الجملة التكرارية التالية Dim i as Integer For i = 1 to 10 ' اكتب الأمر البرمجي هنا Next	لتنفيذ أمر برمجي لعدد معين من المرات (وليكن عشر مرات)
قم بتحديد نقطة البداية والنهاية ثم اكتب كلمة Step واكتب بعدها مقدار الخطوة كما في المثال التالي حيث مقدار الخطوة = 2: Dim i as Integer For i = 2 to 8 step 2 ' أمر برمجي Next	لتحديد خطوة معلومة عند التكرار
تأكد أن الشرط الذي حددته لإنهاء التكرار شرط منطقي وممكن الحدوث.	تجنب التكرار اللانهائي
نستخدم الكلمة Dim لتعريف المتغير وبعد كتابة اسم المتغير وتحديد نوعه نستخدم المعامل = لتعيين قيمة المتغير كما في المثال التالي: Dim x As Single = 23	تعريف المتغير وتعيين قيمته في نفس الوقت
لإنهاء التكرار For...Loop حتى قبل تحقق الشرط نستخدم الكلمة Exit كما في المثال: Dim InpName As String	لإنهاء التكرار For...Loop

	Dim i As Integer For i = 1 To 10 InpName = InputBox("Name?") If InpName = "Trotsky" Then Exit For TextBox1.Text = InpName Next في المثال أعلاه سيتم التوقف عن التكرار عندما يكون الاسم المدخل إلى صندوق المدخلات InputBox هو "Trotsky" حتى إذا لم يتم التكرار لعشر مرات.
تنفيذ أمر برمجي حتى يتحقق شرط معين	نكتب الشرط البرمجي بين Do While و Loop كما في المثال: Dim Query As String = "" Do While Query <> "Yes" Query = InputBox("Trotsky?") If Query = “Yes” Then MsgBox("Hi") Loop في المثال أعلاه سيقوم البرنامج بتنفيذ الأمر البرمجي حتى يتحقق الشرط وهو إدخال الكلمة Yes في تلك الحالة فقط سيظهر لنا صندوق حوار "Hi" وكذلك نستطيع استخدام Do...Loop Until لتحقيق نفس الغرض.
التكرار لمدة زمنية	نستخدم الأداة Timer (المؤقت).
إدخال Snippet في	نضع المؤشر في منطقة الكود في المكان الذي نريد إضافة Snippet له ثم

برنامجنا	نذهب إلى قائمة Edit ونختار منها IntelliSense ثم نختار منها Insert Snippet أو نضغط RightClick في المكان المراد إضافة الــ Snippet له في منطقة الكود ثم نختار Insert Snippet ونبحث عن الــ Snippet المراد ونختاره.
التعرف على الــ Snippets	من قائمة Tools نختار Code Snippets Manager أو من الكيبورد Ctrl+K+B ونتعرف على الــ Snippets المتوفرة مع شرح لكل منها.

الفصل الثامن: مراجعة الأخطاء(Debugging) في البرامج المصممة بالفيجوال بيسك

في الفصول السبعة السابقة قمت بتصميم العديد من التطبيقات أو البرامج وكان لديك الكثير من الوقت لتقع في العديد من الأخطاء (طبعاً أكيد ما قصرت في جانب الأخطاء)، في هذا الفصل ستتعرف على كيفية مراجعة الأخطاء Debugging وعلى كيفية تنفيذ البرنامج للأوامر البرمجية وكيفية التعامل مع الأخطاء المتوقعة. وستتعرف كذلك على أنواع الأخطاء وكيفية معالجتها.

استكشاف وتصحيح الأخطاء:

بعد تصميم برنامجك قد تقع في بعض الأخطاء والتي قد لا تستطيع اكتشافها حتى إذا قمت بمراجعة بسيطة لكافة الأوامر البرمجية التي كتبتها في محرر الكود وحتى إذا راجعت خصائص المكونات التي قد تكون عدلتها، ما الحل في مثل هذه الظروف، فيجوال بيسك 2008 يوفر لك أدوات لمراجعة وتحديد الأخطاء وتصحيحها ضمن أدوات بيئة التطوير، طبعاً هذه الأدوات لن تمنعك من الوقوع في الخطأ، ولكنها تسهل عليك مراجعة الخطأ وتصحيحه.

أنواع الأخطاء الثلاثة:

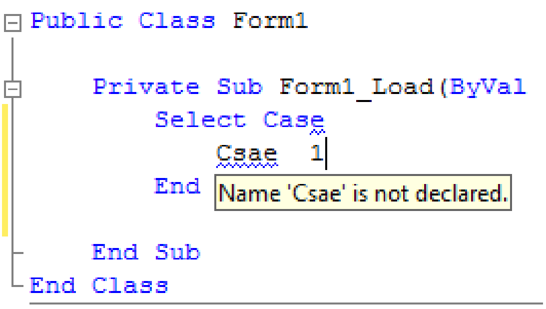
يمكن تقسيم أنواع الأخطاء في الفيجوال بيسك إلى ثلاثة أنواع وهي أخطاء في الأوامر البرمجية وتسمى Syntax Errors، أخطاء وقت تشغيل البرنامج Run-Time Errors، والأخطاء المنطقية Logic Errors.

- الأخطاء في الأوامر البرمجية Syntax Errors وهي الأخطاء التي تحدث في بنية الجمل البرمجية، مثل (الأخطاء المطبعية) الخطأ في كتابة خاصية أو متغير ما (في الـــ Spelling)، لكن الفيجوال بيسك سوف يقوم بتلوين الخطأ مباشرة ولن يسمح لك بتنفيذ البرنامج إذا لم تقوم بتصحيح الخطأ.

- الأخطاء وقت تشغيل البرنامج Run-Time Errors وهي الأخطاء التي تحدث بعد تنفيذ البرنامج وفي وقت تشغيله وتسبب للبرنامج التوقف عن العمل، هذه الأخطاء تحدث عندما يحدث أمر غير متوقع وقت البرمجة أو قد يكون هناك خطأ في الأوامر من النوع الأول Syntax Errors يجبر هذا الخطأ البرنامج على التوقف عن العمل، فمثلاً قد تخطئ في كتابة مسار قاعدة البيانات أو مسار صورة سيقوم البرنامج بوضعها في صندوق الصورة، أو قد تأمر البرنامج بفتح الفلوبي Floppy وحيث لا يوجد فلوبي في الجهاز. هذه الأخطاء تسمى أخطاء وقت التشغيل أو Run-Time Errors وقد توقف البرنامج عن العمل.

- الأخطاء المنطقية Logic Errors وهي أخطاء يكون سببها المبرمج وهذه الأخطاء قد تتسبب بظهور نتيجة خاطئة بسبب أخطاء في البرمجة، فمثلاً قد يحدث خطأ في حفظ إجمالي الفاتورة لإن المبرمج لم يتنبه إلى مسألة الخصم، ومعظم عمليات معالجة الأخطاء تختص بهذا النوع من الأخطاء وطرق معالجتها.

إذا واجهتك أخطاء من النوع الأول فيمكنك حلها بالاستعانة بملفات التعليمات المرفقة مع لغة البرمجة الفيجوال بيسك (Documentation) لتعرف عن الخطأ الذي حدث معك وعن الخصائص وعن الكائنات وكذلك الطرق Methods التي استخدمتها وهل تم التعامل مع واحده منها بشكل خاطئ. خلال عملية كتابة الكود في محرر الكود وإذا حدث أية خطأ يقوم محرر الكود بإضافة خط متعرج تحت الكود الخاطئ، تستطيع معرفة معلومات عن الخطأ بواسطة تحريك الماوس فوق مكان الخطأ. في الصورة التالية ستلاحظ كيف يقوم الفيجوال بيسك بتحديد الخطأ المطبعي عند كتابة Case فبالخطأ كُتبت Csae وبعد أخذ الماوس فوق الكلمة التي تحتها خط انظر ماذا تظهر من رسالة لنا:



فالرسالة في المستطيل ذو اللون الأصفر وتقول Name ‘Csae’ is not declared (وتعني الكلمة Csae غير معرفة).

ملاحظة:

اللون الأخضر تحت الكلمة يعني تحذير، اللون الأحمر يعني أن هناك خطأ في الأمر البرمجي Syntax Error، اللون الأزرق يعني أن المترجم الـــ Compiler حدد الخطأ، أما اللون الأرجواني Purple فيعني أن هناك خطأ آخر.

إذا صادفت خطأ وقت التشغيل Run-Time Error غالباً ما يكون هذا الخطأ بسبب خطأ مطبعي في الـــ Spelling ، إذا كانت الطباعة صحيحة قد يحتاج الخطأ لتعمق أكثر في الأسطر البرمجية التي كتبتها في برنامجك. قد نحتاج لعمل أو لبذل جهد أكبر من هذا بواسطة تجميع الكود على شكل مجموعات ثم تقرير الخطأ من مجموعة معينة، سنتعرف على هذه الطريقة في الفصل التاسع من هذا الكتاب.

تحديد الأخطاء المنطقية Logic Errors

الأخطاء المنطقية في البرنامج تعتبر من أصعب الأخطاء عند التصحيح، وذلك بسبب طبيعتها المنطقية حيث تستلزم المراجعة المنطقية لكل عناصر الكود، تذكر بأن الفيجوال بيسك ليس السبب في مثل هذه الأخطاء ولكن الخطأ من عند المبرمج. لنأخذ هذا المثال:

```
Dim Age As Single
If Age > 13 And Age < 20 Then
   TextBox2.Text = "أنت في عمر المراهقة"
```

```
Else
    TextBox2.Text = "لست في عمر المراهقة"
End If
```

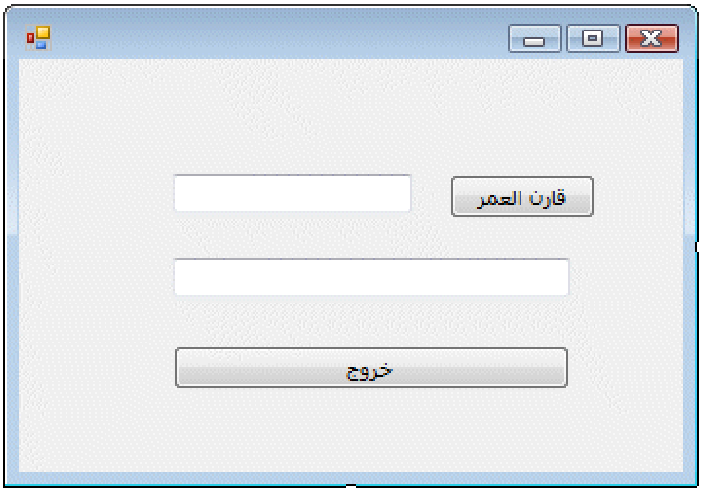
عند مراجعة الكود أعلاه وبما أننا نعرف أن عمر المراهقة يكون من عمر 13 حتى 20 سنة من عمر الشخص، فإن الجملتين أعلاه صحيحتين لكن ماذا إذا أدخلنا العمر 13 أي رسالة ستظهر لنا، بالطبع ستظهر لنا الرسالة التي تقول "لست في عمر المراهقة" مع إن العمر 13 هو من عمر المراهقة، طيب! ما هي المشكلة أو الخطأ، الخطأ هنا أنه لا بد من تحديد العمر بالإشارة >= أكبر أو تساوي 13 وليس بالإشارة > أكبر من 13 فقط، انظر تفاهة هذا الخطأ، كن على ثقة أن معظم الأخطاء البرمجية تأتي من الأخطاء المنطقية ومن جمل مثل هذه (معظم الأخطاء) لكن ليس من السهولة بمكان كشف الأخطاء المنطقية مثل هذه المرة.

مراجعة الأخطاء:

من ضمن الطرق المتاحة لمعرفة الخطأ المنطقي هي التعرف على الكود أو تنفيذ الكود سطر سطر، أي يقوم البرنامج بتنفيذ السطر الأول من البرنامج ثم الذي يليه وإذا حدث الخطأ عرفنا السطر الذي تسبب بالخطأ.

مثال على كيفية مراجعة الأخطاء في برنامج ما

لنأخذ الآن مثالاً فيه الكود السابق، الكود الخاص بالعمر، نفتح الفيجوال 2008 ثم ننشئ مشروع جديد ونسميه debug test ثم نضيف اثنين زر Button واثنين صناديق للنص TextBox (لمعرف كيفية التعديل على الخصائص وكيفية إضافة المكونات قم بمراجعة الدروس السابقة أو الصفحات السابقة من هذا الكتاب) ونعدل الخصائص ليكون الفورم بالشكل التالي:



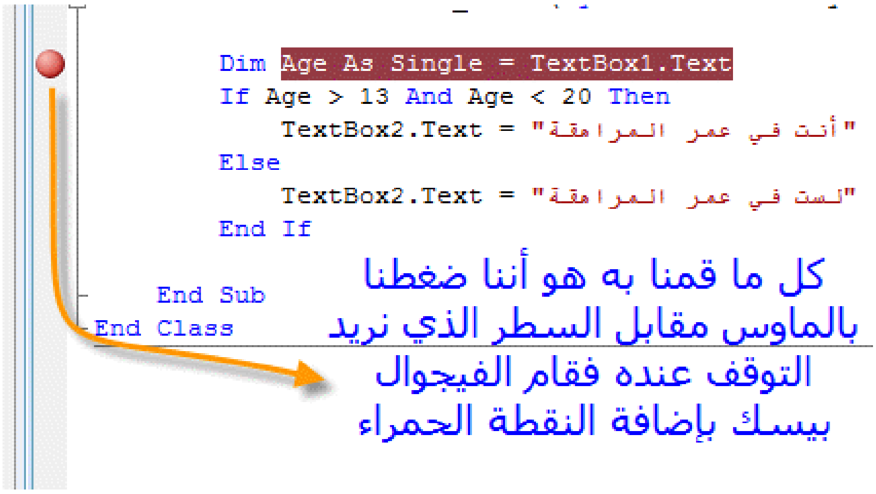
نضغط Double-Click على زر "خروج" ونكتب كود إغلاق البرنامج وهو

```
End
```

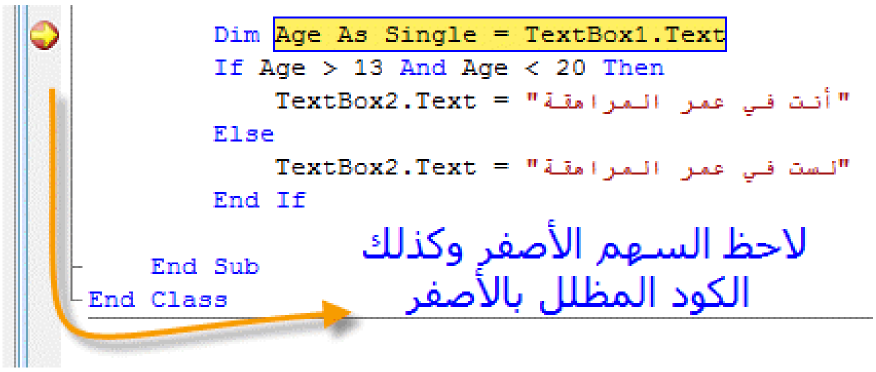
ونضغط على الزر "قارن العمر" ونكتب الكود التالي فيه:

```
Dim Age As Single = TextBox1.Text
If Age > 13 And Age < 20 Then
    TextBox2.Text = "أنت في عمر المراهقة"
Else
    TextBox2.Text = "لست في عمر المراهقة"
End If
```

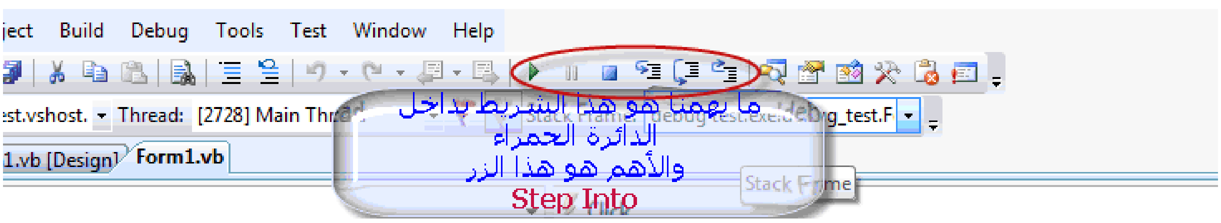
بعد تنفيذ البرنامج وكتابة العمر 20 في صندوق النص ثم الضغط على الزر "قارن العمر" تظهر لنا رسالة Message Box تقول لنا "لست في عمر المراهقة" أما إذا كتبنا العمر 15 فتظهر رسالة تقول "أنت في عمر المراهقة"، طيب الآن ماذا إذا أدخلنا العمر 13 ستظهر لنا رسالة تقول "لست في عمر المراهقة" مع إن العمر 13 من عمر المراهقة. الآن دعونا نضيف نقطة توقف أو ما يسمى بـــ Break Point إلى منطقة الكود إلى الكود = Dim Age As Single TextBox1.Text كما في هذه الصورة:



نقوم الآن بتشغيل البرنامج ونكتب العمر 16 ثم نضغط الزر "قارن العمر" نلاحظ أن البرنامج ينقلنا إلى محرر الكود ليبين لنا أن هناك نقطة توقف عند الكود المراد تنفيذه(أما إذا كان الكود عند نقطة التوقف Break Point لا علاقة له بالعملية التي يقوم بها البرنامج فلن يتوقف(إنما يتوقف البرنامج فقط عند نقطة التوقف إذا كانت العملية المناطه بالبرنامج لها علاقة بالكود المحجوز في نقطة التوقف))، الآن نحن أما محرر الكود بل إن البرنامج يحجز الكود عند نقطة التوقف باللون الأصفر ليبين لنا بأن هذا الكود المعني بتنفيذ العملية التي طلبتها من البرنامج، لاحظ الصورة:



الآن بيئة التطوير متوقفة عن تنفيذ الأمر البرمجي، ولتنفيذ الأمر نذهب إلى الأدوات أعلى الفورم انظر الصورة:



من داخل الدائرة الحمراء في الصورة أعلاه نختار الزر Step Into، ليقوم الفيجوال بيسك بتنفيذ الأمر البرمجي وبسبب أن الكود مترابط مع بعضة البعض سينتقل اللون الأصفر إلى السطر التالي ويتوقف، نختار Step Into مرة ثانية، سينتقل اللون الأصفر إلى السطر الذي يليه، مع ملاحظة أن السطر المظلل باللون الأصفر لا يقوم البرنامج بتنفيذه إلا بعد الضغط على Step Into، نقوم بالضغط على Step Into مرة ثالثة، الآن وبما أن العمر الذي وضعناه في صندوق النص هو 16 أي أنه يتوافق مع الشرط الأول في جملة If الشرطية، لذلك لن يقوم البرنامج بالتحقق من الشرط الثاني وإنما سيقوم بالانتقال مباشرة إلى End If. وإذا ضغطنا Step Into للمرة الرابعة نلاحظ أن اللون الأصفر ينتقل إلى End Sub وعند الضغط على Step Into للمرة الأخيرة نلاحظ أن البرنامج يقوم بإظهار الرسالة "أنت في عمر المراهقة" في صندوق النص المحدد لذلك. لعلنا الآن قد عرفنا كيف يتسلسل التطبيق تطبيق الكود البرمجي في برامجنا. دعونا نعيد العملية السابقة مرة أُخرى وبعد وضع العمر 16 والضغط على الزر "قارن العمر" وعند توقف اللون الأصفر فوق الكود:

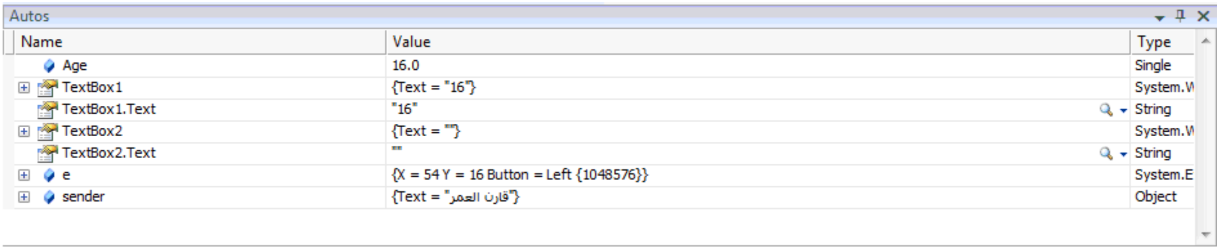
```
Dim Age As Single = TextBox1.Text
```

نذهب الماوس إلى فوق المتغير Age نشاهد الكتابة التالية تحت مؤشر الماوس (Age |0.0) لماذا وماذا تعني هذه القيمة، للتوضيح مادمت في وضع الـــ Debugging إذا وضعت الماوس فوق أي متغير من المتغيرات على طول الكود فسيكتب لك محرر الكود كم هي قيمة المتغير في تلك اللحظة، وقد وضع القيمة 0.0 للمتغير Age لإن المتغير حتى هذه اللحظة لم تتم تعبئته بالقيمة الموجودة في صندوق النص TextBox1.Text ، هذه الميزة ستستفيد منها في البرامج العملاقة والمعقدة، لنعرف متى يتم تعبئة المتغير بالقيمة المحددة له.

ونلاحظ أننا بعد الضغط على Step Into للمرة الثانية ثم نضع الماوس فوق المتغير Age نلاحظ ظهور الكتابة (Age |16.0) فوق الماوس لتعلمنا بأنه قد تمت تعبئة المتغير Age بالقيمة 16 عند الانتقال من السطر الأول في الكود إلى الذي يليه.

بالاستمرار بالضغط على Step Into نلاحظ انتقال اللون الأصفر الذي فوق الكود من سطر إلى آخر لنعرف أي سطر من سطور الكود يتم تنفيذه.

لنذهب أبعد مع مراجعة الأخطاء أو مع الـــ Debugging فمن قائمة بيئة التطوير نختار القائمة المنسدلة Debug ثم نختار منها Windows ثم نختار منها Autos لنلاحظ ظهور نافذة طافية بالأسفل لـــ Autos هذه النافذة الصغيرة تظهر لنا العديد من المعلومات التي قد نستفيد منها في مراجعة الكود من ضمن هذه المعلومات قيمة المتغير عند كل سطر من سطور الكود وكذلك جميع المكونات (التي تتغير قيمتها في هذه اللحظة) والموجودة على الفورم دعونا نلاحظ هذه الصورة لنافذة Autos:



نلاحظ في الصورة أعلاه المتغير Age وقيمته عند الانتقال من سطر إلى آخر وكذلك قيمة المكونات المختلفة على الفورم مثل مربعات النص TextBox1, TextBox2 وحتى بقية المكونات مثل e, sender ومع الاستمرار بالضغط على Step Into سنلاحظ تغير قيمة كل عنصر من المكونات أعلاه.

لنجرب فتح البرنامج ومحاولة معالجة الأخطاء بوضع قيمة العمر 13 ثم ملاحظة النتيجة التي تظهر في TextBox2 وهي "لست في عمر المراهقة" ثم لنضع العمر 20 نلاحظ نفس النتيجة. نستنتج بأن هناك خطأ ما في بنية البرنامج وبالتحديد في تركيب الجملة الشرطية حيث أن الكود:

```
If Age > 13 And Age < 20 Then
```

فيه خطأ لإن الشرط فيه أن يكون العمر أكبر من 13 وأصغر من 20 والصحيح أن يكون العمر أكبر من أو يساوي 13 وأصغر من 20. كالتالي :

```
If Age >= 13 And Age < 20 Then
```

ولذلك لا بد لك من تعديل الكود المذكور أعلاه (وللعلم توجد نسخة من التطبيق أعلاه في المرفق رقم 023 لكن لابد عليك من تعديل الكود بعد تطبيق درس مراجعة الأخطاء عليه).

متابعة المتغيرات باستخدام Watch Window

لاحظنا في المثال أعلاه أهمية استخدام الأداة Autos المتوفرة ضمن قائمة Debug في بيئة التطوير وكيف أن هذه الأداة مفيدة جداً وهامة في متابعة قيم المتغيرات على الفورم وقت الانتقال من سطر إلى آخر على الكود فأداة Autos تظهر لنا جميع القيم للمتغيرات والمكونات التي تغيرت عند تطبيق سطر معين في الكود (وهو السطر المظلل باللون الأصفر). عندما يقوم البرنامج بتنفيذ أكواد لا علاقة لها بالمتغيرات فإن هذه القيم لا تظهر في قائمة Autos لذلك فنحتاج إلى أداة تظهر لنا هذه القيم، إنها أداة Watch Window تقوم هذه الأداة بمتابعة القيم والمتغيرات الهامة في الكود مادمت في مرحلة مراجعة الأخطاء الـــ Debugging Mode في فيجوال بيسك 6 كان بإمكاننا فتح نافذة Watch Window واحدة أما في فيجوال 2008 فبإمكاننا فتح أربع نوافذ مرقمة 1، 2، 3، 4. لإضافة Watch Window لا بد أن يكون التطبيق في مرحلة الـــ Debugging نختار Debug ثم Windows ثم Watch ثم نختار واحدة من الأربع. ونستطيع إضافة تعبير معين لها لمراقبته فمثلاً يمكننا إضافة التعبير Age >= 13 لتقوم نافذة الـــ Watch Window بمراقبة المتغير Age >= 13.

المرفقات التي نقوم بالتطبيق عليها في هذا الباب حتى الآن موجودة بالمرفق رقم 023.

بعد تشغيل التطبيق في المرفقات رقم 023 والتأكد من أن الـــ Brake Point عند السطر :

Dim Age As Single = TextBox1.Text نقوم بتنفيذ البرنامج بالضغط على F5 ثم نقوم بكتابة الرقم 13 في الفورم التي ستظهر ثم نضغط الزر "قارن العمر" وحين يتوقف الفيجوال بيسك 2008 عند نقطة التوقف الـــ Brake Point نذهب إلى منطقة الكود ثم نختارStep Into لنقوم بتظليل المتغير المراد مراقبته ثم الضغط Right-Click على المتغير ثم اختيار Add Watch ستقوم بيئة التطوير مباشرة بإضافة المتغير إلى الـــ Watch Window فمثلاً نظلل المتغير Age ونختار Add Watch وكذلك TextBox2 وبعد إضافتهم نختار Step Into أو نضغط على F8 لنقوم بنفس العمل ماذا نلاحظ سنلاحظ بأن بيئة التطوير قد قامت بتغيير القيمة

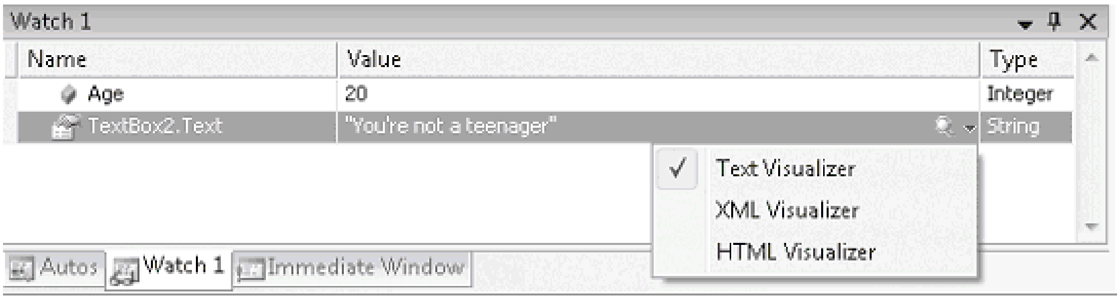
للمتغير Age إلى 13 (التي كتبناها في صندوق النص في الفورم) وكذلك لونت القيمة باللون الأحمر لننتبه بأن هذه القيمة تغيرت في هذه اللحظات. نستطيع إضافة أي بند إلى Watch Window بعمل Right-Click على البند ثم اختيار Add Watch. ويمكننا حذف بنود الـــ Window Watch باختيار أحد البنود ثم الضغط على Delete.

عناصر التكبير أو Visualizers

استخدمنا سابقاً العديد من الوسائل التي تعيننا على معرفة قيمة المتغيرات ومتابعتها على طول الكود لمراجعة الأخطاء التي قد تكون في البرنامج وكذلك لمعرفة كيف تتم تعبئة المتغيرات بقيمها وكيف تتعامل مع بعضها في منطقة الكود. الآن سنستخدم ما يسمى بالمكبرات أو الـــ Visualizers (ستلاحظ ركاكة ترجمة بعض المصطلحات للعديد من الأسباب لذلك فقد تم إرفاق الكلمة الإنجليزية الأصلية حتى لا يحدث لبس) ونستخدم عناصر التكبير لأسباب معينة ففي السابق كنا نستخدم Windows Watch و Autos لمتابعة قيم المتغيرات ولكن ماذا إذا كان لدينا متغير يعرض لنا سيل من البيانات مثل المتغير الذي يعرض لنا قيمة معينة من قاعدة بيانات أو يعرض لنا نص معين من صفحة في الإنترنت عندئذ لابد علينا من استخدام عناصر التكبير أو Visualizers وتسمى عناصر التكبير بسبب الإيقونة التابعة لها وهي عبارة عن عدسة تكبير.

لدينا ثلاثة أنواع من عناصر التكبير Text Visualizer و XML Visualizer و HTML Visualizer ويبين كل نوع منها نوع البيانات المخرجة هل هي Text أو XML أو HTML.

لنأخذ مثال على عناصر التكبير حيث يفترض بنا أننا الآن فاتحين المثال debug test الموجود ضمن المرفقات برقم والذي استخدمناه في التمارين السابقة. ويفترض الآن بأننا في مرحلة الـــ Debugging وفاتحين الـــ Window Watch وخلال عملية الضغط على Step Into نلاحظ تَكَون عدسة صغيرة في داخل الـــ Window Watch في عمود Value وبجانبها سهم صغير كما في الصورة:

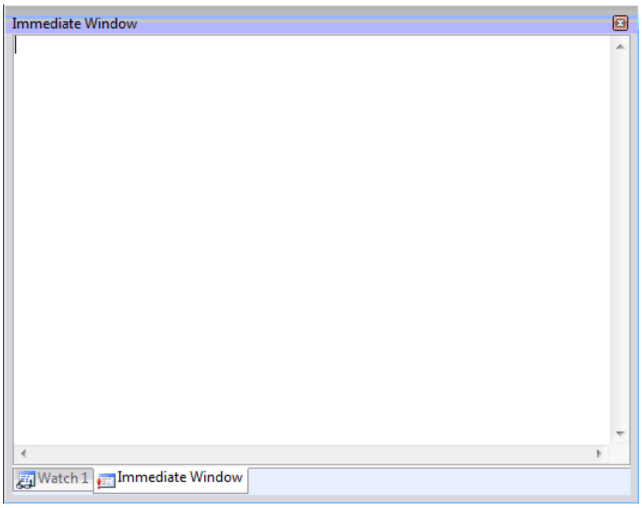


(الصورة من كتاب المؤلف لإن العدسة لم تتكون عندي لكني وجدت هذه العدسة في منطقة الكود وبجانب المتغيرات أو الـــ TextBox فبعد وضع الماوس فوق كلمة TextBox2 في الكود ظهرت العدسة وتم اختيار Text Visualizer لتظهر لنا فورم صغير فيه قيمة النص الموجود بداخل صندوق النص وعند اختيار XML Visualizer ظهرت نافذة صغيرة تظهر لنا النص الموجود في صندوق النص TextBox2 مع وصلة لملف XML في ملف الـــ Temp فيه تفاصيل النص الموجود في صندوق النص ولكن بصيغة XML).

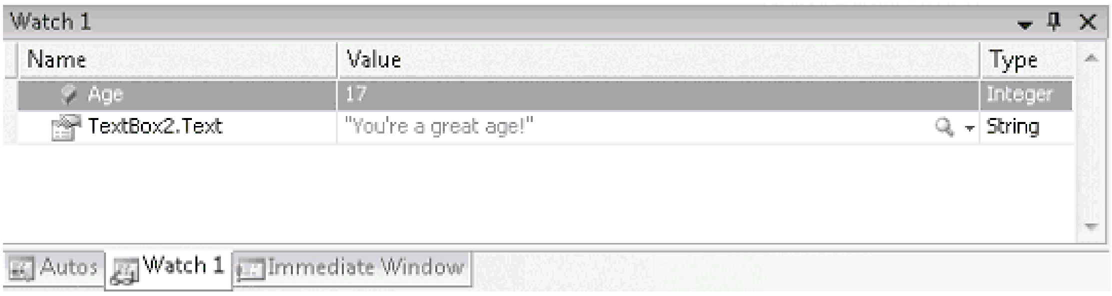
النافذة السريعة Immediate Window

(ملاحظة للمرة الثانية بأن ترجمة المصطلحات قد لا تكون صحيحة 100% لذلك تم إرفاق المصطلح الأصلي) لحد الآن تم استخدام العديد من الأدوات المتوفرة ضمن بيئة التطوير لمراجعة قيم المتغيرات والعديد من العناصر ضمن الكود لكننا هنا بصدد أداة تختلف بعض الشيء عن الأدوات السابقة وتعتمد على الأدوات السابقة اعتماداً كلياً وبخاصة الـــ Window Watch ونستخدم النافذة السريعة أو Immediate Window لإدخال قيم معينة على المتغيرات أو في صناديق النص أو في أي عنصر ضمن الكود بدون التأثير المباشر على التطبيق أو البرنامج الذي تم تصميمه فمثلا في المثال debug test يمكننا تعيين قيمة للمتغير Age وكذلك كتابة قيمة معينة في صندوق النص TextBox2 وملاحظة أثار ذلك في Window Watch بدون التأثير على البرنامج.

لنأخذ المثال السابق وفي مرحلة الـــ Debugging نذهب في بيئة التطوير إلى القائمة Debug ثم Windows ثم Immediate سنلاحظ ظهور نافذة جديدة تسمى Immediate Window رديفة وملاصقة للنافذة Window Watch كما في الصورة:



في نافذة الـــ Immediate Window قم بتعيين قيمة للمتغير Age من خلال كتابة Age = 17 ثم الضغط على Enter ثم قم بفتح النافذة الظاهرة في خانة Watch1 ماذا تلاحظ، ستلاحظ ظهور قيمة المتغير Age فيها بـــ 17 . انظر الصورة (من الكتاب الأصلي):



(بصراحة عندي لم تظهر الرسالة التي يشير إليها المؤلف فمن وجد الحل لهذه المعضلة فليراسلني بالإيميل وليوضح الحل في منتدى فيجوال بيسك العرب).

الانتقال من النافذة السريعة Immediate Windowإلى نافذة الأوامر Command Window:

إذا أردنا الانتقال من النافذة السريعة إلى نافذة الأوامر نكتب الأمر cmd< في النافذة السريعة أو الأمر Immed< في نافذة الأوامر.

ونستفيد من نافذة الأوامر في العديد من المهام والتي هي التعامل مع الفيجوال بيسك بواسطة سطر الأوامر (من يتذكر نظام الـــ DOS) فمثلاً كتابة الأمر File.SaveAll يقوم بحفظ التطبيق الحالي

الذي نعمل عليه، وللعلم فإن نافذة الأوامر تحتوي على تعبئة تلقائية Auto-complete فبمجرد كتابة بعض أحرف من الأمر البرمجي تقوم هذه النافذة بعرض قائمة من المقترحات لاختيار أحدها إذا أردنا تسهيل الأمر أو إذا نسينا الـــ Spelling.

التطبيق الذي قمنا بالعمل علية في هذا الباب موجود في المرفقات برقم 023.

خطوة للأمام (إلغاء نقاط التوقف) Breakpoints:

خلال هذا الفصل تعرفنا على كيفية إضافة نقاط التوقف Breakpoints وكيفية الاستفادة منها على طول الكود وهنا سنتعرف على كيفية إزالتها من الكود.

نذهب إلى محرر الكود و بالماوس نقوم بالنقر فوق الدائرة الحمراء والتي تعني نقطة توقف فتزول هذه النقطة لتنتهي من تلك المنطقة نقطة التوقف. وإذا كان لديك العديد من نقاط التوقف في منطقة الكود فمن الصعب استخدام هذا الطريقة ويفضل الذهاب إلى قائمة Debug واختيار Delete All Breakpoints أو بالضغط على Ctrl+Shift+F9 ليتم إلغاء كافة نقاط التوقف.

خلاصة الفصل الثامن:

من اجل أن	**قم بالتالي**
إظهار صندوق أدوات مراجعة الأخطاء	من قائمة View اختر Toolbars ومنها Debug
وضع نقطة توقف Breakpoint	في محرر الكود وعند الجملة أو الأمر الذي تريد التوقف عنده قم بالنقر على الماوس في الهامش المقابل للجملة من اليسار، وعند تنفيذ البرنامج سيقوم البرنامج بالتوقف عند وصولة إلى تلك الجملة.
تنفيذ سطر برمجي واحد ضمن الكود	الضغط على StepInto ليتم تنفيذ سطر برمجي واحد فقط وعند الضغط ثانية يتم تنفيذ السطر الذي بعده وهكذا.

معرفة قيمة متغير ما وقت معين خلال تشغيل البرنامج	الذهاب بالماوس إلى منطقة الكود ووضع الماوس فوق المتغير المراد معرفة قيمته حينها ستقوم بيئة التطوير بإظهار قيمته.
استخدام نافذة الـــ Autos لمعرفة قيمة متغير معين	في مرحلة تنفيذ البرنامج الـــ Debugging نذهب إلى قائمة Debug ثم Windows ثم Autos.
إضافة قيمة متغير أو خاصية معين/ه إلى الـــ Watch Window	في مرحلة تنفيذ البرنامج الـــ Debugging نختار القيمة المرادة في محرر الكود ثم Right-Click ثم Add Watch.
إظهار Watch Window	في مرحلة تنفيذ البرنامج الـــ Debugging انقر قائمة Debug ثم Windows ثم Watch ثم نختر واحدة من الأربع.
إظهار بيانات المتغيرات وغيرها بنوعية HTML أو XML	في نافذة الـــ Autos اختر Visualizer ثم Watch ثم Right-Click
فتح نافذة Immediate Window	من قائمة Debug نختار Windows ثم Immediate
كتابة أمر من خلال نافذة الأوامر	في نافذة الأوامر نكتب الأمر الذي نريد فمثلا الأمر File.SaveAll يقوم بحفظ التطبيق الذي نعمل عليه.
التنقل بين Immediate	في Immediate Window نكتب >cmd ونضغط على Enter وفي

Window و Command Window	Command Window نكتب >immed ونضغط على Enter.
إلغاء نقط التوقف Breakpoints	في محرر الأكواد نضغط فوق الدائرة الحمراء أو من قائمة **Debug** نختار Delete All Breakpoints أو نضغط على Ctrl+Shift+F9

الفصل التاسع صيد الأخطاء باستخدام البناء الهيكلي Structure Error Handling

في الفصل الثامن تعرفنا على كيفية مراجعة الكود وكيفية تعامل بيته التطوير مع المتغيرات والخصائص خلال مرحلة تنفيذ البرنامج وكذلك على وضع نقاط التوقف ومراقبة كيفية التعامل معها من قبل البرنامج، في هذا الفصل سوف نتعرف على كيفية التعامل مع الأخطاء بطريقة سلسة ومنظمة وهيكلية ولنبدأ معاً:

صيد الأخطاء بواسطة الجملة Try ... Catch Statement

إذا حدث أية خطأ صغير وقت البرمجة قد يؤثر بشكل كبير جداً على مرحلة تنفيذ البرنامج فمثلاً إذا لم يستطيع البرنامج فتح ملف معين قد حُدد له سلفاً فإنه قد ينهار ويقوم الويندوز بإغلاقه. ففشل البرنامج في فتح ملف أو في تحميل صورة أو في التعامل مع أرقام أو نصوص أو مدخلات قد يخطئ المستخدم فيها مثل الطلب من البرنامج أن يقسم ويكون مقام القسمة صفر، هذه الأخطاء وغيرها والتي تعتبر تافهة في تنظرنا نحن البشر قد تكون معضلة وداهية في نظر الكمبيوتر المسكين وذلك لإن الكمبيوتر لا يفكر وإنما يقوم بما أُمر به فقط من قبل المبرمج وحدوث مثل تلك الانهيارات للبرنامج يعود السبب فيها للمبرمج الذي لم يحدد للبرنامج ماذا يفعل إذا حدث له خطأ ما.

وبدلاً من كتابة جميع أنواع الأخطاء المحتملة للبرنامج ليقوم بالتعامل معها يمكننا أن نكتب جملة واحدة نحدد فيها للبرنامج فيما إذا حدث مشكلة ما في البرنامج ماذا يفعل. قد نضع هذه الجملة في كل إجراء يتوقع حدوث خطأ فيه بحسب الحاجة.

متى نستخدم جُمَل صيد الأخطاء:

باختصار يجب علينا وضع جُمَل صيد الأخطاء في كل جملة برمجية متوقع أن يحدث فيها خطأ ما وبشكل عام تكون هذه الأخطاء بسبب مؤثرات خارجية من خارج البرنامج وليس من داخله فمثلا الخطأ الذي قد يوجد إذا لم يكن هناك اتصال بالانترنت أو بسبب تعطل الشبكة الداخلية أو بسبب عدم وجود سي دي أو بسبب عدم توصيل الطابعة كل هذه الأخطاء قد تسبب لنا العديد من المشاكل إذا لم نتحاشاها بجُمَل صيد الأخطاء.

وباستخدام الجملة Try...Catch نستطيع تصيد الأخطاء حيث يتم وضع الكلمة Try قبل الجملة التي نتوقع حدوث خطأ بسببها ووضع Catch مباشرة مع قائمة بالأوامر التي يجب على البرنامج القيام بها إذا حدث الخطأ. وبإمكاننا استخدام العديد من الكلمات الوسيطة مع Try و Catch مثل Catch When و Finally و Exit Try والآن لننظر إلى الصياغة العامة في جملة Try...Catch كالتالي:

Try

الجملة البرمجية التي قد يحدث بسببها خطأ

Catch

الجملة التي نريد تنفيذها حال حدوث الخطأ

Finally (اختياريه)

جملة أخيرة نريد تنفيذها سواء جدث الخطأ أم لم يحدث

End Try

الآن دعونا نتخيل الجمل البرمجية فبعد كلمة Try نضع الجملة البرمجية التي نحن متخوفين من أنها قد تحدث خطأ ما في البرنامج مثل جملة فتح ملف في مسار ما قد لا يوجد هذا المسار أو قد يكون الملف معطوباً أو غيرها من الأخطاء، ونضع الجملة التي نريد تنفيذها إذا حدث خطأ بعد كلمة Catch وفي الأخير نضع الجملة التي نريد فيها إنها عملية فتح الملف سواءاً فتح الملف أو لم يفتح بعد الكلمة Finally.

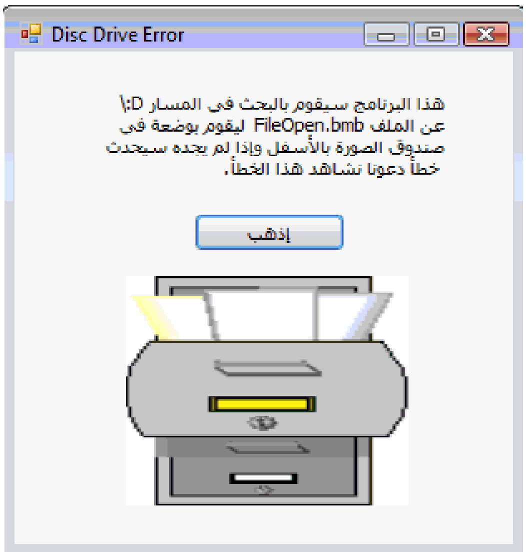
مثال تطبيقي على التعامل مع المسارات والملفات:

لنأخذ الآن مثال فيه احتمال على حدوث خطأ ما هذا المثال هو التعامل مع الملفات والمسارات حيث سنصمم فورم فيه زر Button ومربع للصورة عند الضغط على الزر سيذهب البرنامج إلى مسار ما وسيقوم بتحميل صورة معينة في مربع الصورة إذا وجدها وإذا لم يجدها فسيحدث الخطأ.

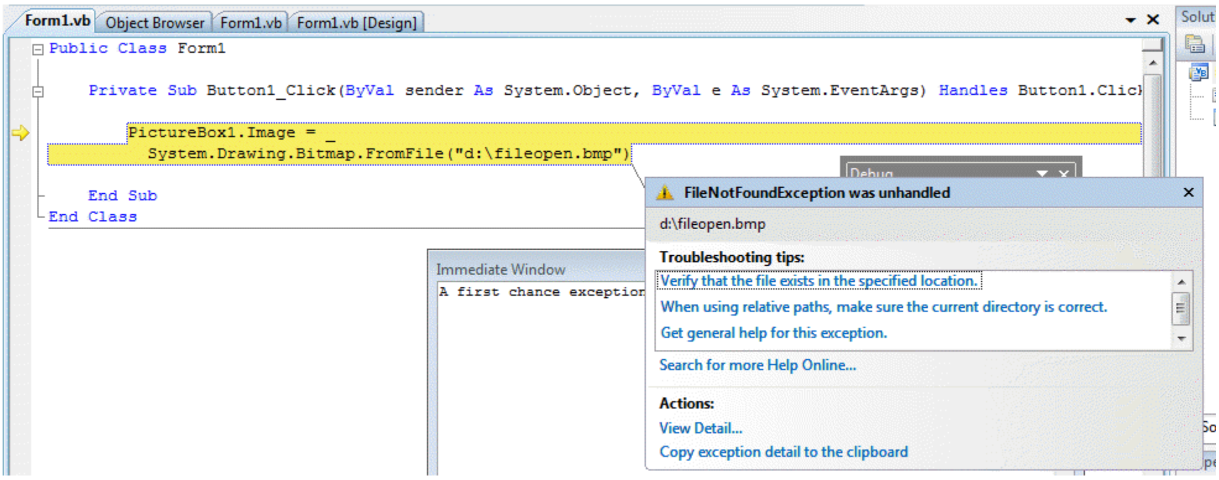
المثال موجود في المرفق رقم 024 نفتح الملف ونأخذ الصورة التي باسم FileOpen.bmp ونضعها في المسار D:\ ثم نقوم بفتح البرنامج بواسطة بيئة التطوير ونذهب إلى محرر الأكواد لنجد الكود :

```
PictureBox1.Image = _
        System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
```

وهذا الكود يجعل البرنامج يقوم بالذهاب إلى القرص D:\ ليبحث عن الملف الصورة fileopen.bmp ويضعها في مربع الصورة PictureBox1 وبما أننا قد وضعنا الصورة في القرص D:\ نقوم بتشغيل البرنامج وضغط الزر ليقوم البرنامج بوضع الصورة في مربع الصورة كما في الصورة التالية:



الآن وبعد أن رأينا كيف قام البرنامج بتحميل الصورة من المسار المحدد نذهب إلى القرص D:\ ونلغي الصورة Fileopen.bmp ونعيد تشغيل البرنامج ونضغط على الزر والآن يحدث الخطأ كما في الصورة (لأن البرنامج لم يجد الصورة في المسار المحدد في الكود):



أما إذا كان البرنامج يعمل بدون الفيجوال بيسك 2008 فسيكون الخطأ كما في الصورة التالية:



وحدث هذا الخطأ بسبب عدم وجود الصورة في المسار المحدد. لنقل الآن بتصحيح الخطأ في الحقيقة الملف الذي لا يوجد في المسار المحدد وهو D:\ لا بد أن يتم وضعه لتصحيح الخطأ لكن الخطأ الذي نقصده هنا هو الخطأ البرمجي وذلك بطرح بدائل للبرنامج يقوم بها إذا لم يوجد الملف في المسار المعين كما في المثال التالي:

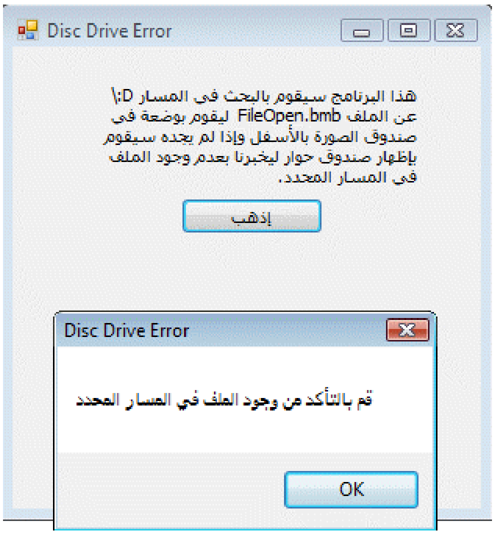
مثال على معالجة الأخطاء باستخدام Try و Catch

في هذا المثال سنستخدم نفس معطيات المثال السابق مع بعض التعديلات في الكود لنتعامل مع خطأ احتمال عدم وجود الملف في المسار المحدد. سنستخدم المثال في المرفق رقم 025.

نفتح المثال لنضع الكود التالي تحت حدث النقر على الزر:

```
Try
   PictureBox1.Image = _
     System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
   MsgBox("قم بالتأكد من وجود الملف في المسار المحدد")
End Try
```

لنجرب الآن تشغيل البرنامج مع عدم وجود الملف fileopen.bmp في D:\ ونرى ماذا سيحدث للبرنامج لنرى الصورة:



الآن بدلاً من انهيار البرنامج فإنه يظهر لنا رسالة تخبرنا بعدم وجود ملف في المسار المحدد في الكود. علينا الآن أن نتعرف على الفوارق بين جُمَل Try...Catch.

استخدام Finally لغرض تحسين الإجراء البرمجي:

كما وضحنا سابقاً إن استخدام Finally هو إجراء اختياري نكتب بعدة الكود الذي نريد تنفيذه سواءاً حدث الخطأ أم لم يحدث، لكن قد يصير مهما للعديد من الأغراض ومنها إنهاء التواصل مع قاعدة البيانات أو تحديث قيمة متغير معين أو خاصية محددة أو إظهار نتيجة معينة أو إعلامنا بانتهاء الإجراءات البرمجية سنقوم في المثال التالي باستخدام Finally ولنأخذ الملف في المرفق رقم 026.

ولنكتب هذا الكود في حدث النقر على الزر:

```
Try
    PictureBox1.Image = _
      System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
    MsgBox("قم بالتأكد من وجود الملف في المسار المحدد")
Finally
    MsgBox("تم إنهاء العملية")
End Try
```

بعد تنفيذ البرنامج وظهر رسالة الخطأ أو حتى مع عدم ظهور رسالة الخطأ (أي عند وجود الملف وتحميله إلى مربع الصورة) سيظهر لنا صندوق الحوار التالي:



جُمَل Try…Catch الأكثر تعقيداً

في الأمثلة السابقة تعاملنا مع الجملة Try…Catch وبشكل مبسط الآن سنتعامل مع جُمَل Try…Catch أكثر تعقيداً لصنع تطبيقات أكثر احترافية.

التعامل مع الكائن Err

للعلم الكائن Err كان متوفراً مع النسخ القديمة من الفيجوال بيسك والنسخ الأقدم لكن الفرق هنا هو أن الكائن في فيجوال بيسك 2008 قد تم تحديثه وإضافة العديد من الميزات له من ضمنها إعطاء تقرير أفضل عن الأخطاء وأرقامها وغيرها من التفاصيل كما يوجد العديد من الكائنات التي تتعامل مع الأخطاء مثل الكائن Exception ذو القدرات العالية في التعامل مع الأخطاء.

من ضمن الميزات التي يحويها الكائن Err هي رقم الخطأ Err.Number ووصف الخطأ Err.Description فرقم الخطأ يحتوي على رقم أخر خطأ حدث ووصف الخطأ يحتوي على وصف قصير لذلك الخطأ وباستخدام الخاصيتين Err.Number و Err.Description يمكننا الحصول على رقم الخطأ ووصف الخطأ ووضع أكواد برمجية للتعامل مع كل خطأ بطريقة معينة أو وضع أكواد للتوضيح للمستخدم ماذا يجب القيام به للتعامل مع مثل تلك الأخطاء. نستطيع مسح تقرير الخطأ في الكائن Err بواسطة الكود Err.Clear في الجدول التالي يوجد أرقام لبعض الأخطاء التي قد تحدث في البرنامج ويقوم الكائن Err بإظهار أرقامها إذا حدثت وللعلم هناك أخطاء معقدة بعض الشيء خاصة في التعامل مع قواعد البيانات ستجدها في ملفات المساعدة المرفقة مع الفيجوال بيسك 2008. أما الأرقام الغير مستخدمة من 1– إلى 1000 فسيتم استخدامها مستقبلاً من قبل مطوري الفيجوال بيسك. جدول الأخطاء:

رقم الخطأ	رسالة الخطأ
5	خطأ في الاستدعاء Procedure call or Argument is not valid
6	الإغراق Overflow
7	أكبر من قوة الذاكرة Out of memory
9	Subscript out of range

القسمة على صفر	11
Type mismatch عدم التطابق النوعي	13
dll Error in loading DLL خطا عند تحميل مكتبة الـــ	48
Internal error خطأ داخلي	51
Bad fi le name or number اسم ملف خاطئ أو رقم	52
File not found لا يوجد الملف	53
File already open الملف حالياً مفتوح	55
Device I/O error	57
File already exists الملف موجود مسبقاً	58
القرص ممتلئ	61
Input past end of file	62
توجد ملفات كثيرة	67
لا يوجد الجهاز	68
لا توجد صلاحية	70
القرص ليس جاهزاً	71
Cannot rename with different drive	74
Path/File access error	75

76	لا يوجد المسار
91	Object variable or With block variable not set
321	نوعية الملف غير صالحة
322	ليس بالإمكان كتابة الملف المؤقت اللازم
380	قيمة الخاصية غير مقبول
381	Property array index is not valid
422	لا توجد الخاصية
423	الخاصية أو الطريقة غير موجودة Property or method not found
424	Object required
429	Cannot create Microsoft ActiveX component
430	Class does not support Automation or does not support expected interface
438	Object does not support this property or method
440	Automation error
460	Clipboard format is not valid
461	Method or data member not found
462	لا يوجد سيرفر
463	Class not registered on local machine
481	الصورة غير صالحة

482	خطأ في الطابعة.

في المثال التالي سنتعرف على كيفية الاستفادة من رقم ووصف الخطأ في الكائن Err.

لنأخذ التطبيق السابق ونكتب الكود التالي في حدث النقر على الزر:

```
Try
   PictureBox1.Image = _
     System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch When Err.Number = 53
   MsgBox("تأكد من الملف ومسار الملف")
Catch When Err.Number = 7
   MsgBox("لا يمكن فتح الصورة بسبب تجاوزها لإمكانيات الذاكرة", , Err.Description)
Catch
   MsgBox("حدث خطأ عند تحميل الملف", , Err.Description)
End Try
```

عُد إلى الكود أعلاه ولاحظ أننا حددنا رقمين من أرقام الخطأ: الرقم 53 (لا يوجد الملف) ستظهر هذه الرسالة إذا لم يوجد الملف، والرقم 7 (عدم كفاية الذاكرة) ستظهر هذه الرسالة إذا حاولنا فتح ملف وورد أو أكسل في داخل مربع الصورة (وذلك بتعديل اسم الملف من fileopen.bmp في الكود إلى اسم ملف وورد أو أكسل) وإذا حدث خطأ آخر غير هذين الخطأين سيقوم البرنامج بإظهار الرسالة الأخيرة وهي "حدث خطأ عند تحميل الملف".

قم بتجربة المثال أعلاه أعثر على الأخطاء: عدم وجود الملف (بحذف الملف من القرص) وعدم كفاية الذاكرة (بتعديل الكود ليقوم البرنامج بفتح ملف وورد أو أكسل في صندوق الصورة (فقط قم بتغيير fileopen.bmp في الكود إلى file.doc أو file.xls ولابد من توفر هذا الملف في القرص ليحدث الخطأ "عدم كفاية الذاكرة")).

يوجد التطبيق في المرفق رقم 027.

باستخدام رقم ووصف الخطأ تستطيع مساعدة المستخدم على فهم طبيعة عمل البرنامج وإيجاد الحلول المناسبة للأخطاء.

أخطاء مع سابق الإصرار والترصد:

للعديد من ألأسباب قد نريد أن نعرف مدى تجاوب البرنامج مع الأخطاء لذلك قد نقوم بإلقاء Raise هذه الأخطاء في برنامجنا والتعرف على كيفية معالجتها من قبل البرنامج فمثلاً نستطيع كتابة هذا الكود في زر "اذهب" في حدث النقر على الزر:

```
Try
   Err.Raise(61)
Catch When Err.Number = 61
   MsgBox("القرص ممتلئ")
End Try
```

فسيتعامل التطبيق وكأنه قد واجه خطأ من النوع ذي الرقم 61 وقد تعامل معه صندوق حوار فيه "القرص ممتلئ".

تحديد عدد مرات المحاولة بـــ Try...Catch

نستطيع تحديد عدد مرات المحاولة بـــ Try...Catch بعدد محدد باستخدام متغير كما في المثال التالي: (التطبيق موجود في المرفقات برقم 028.)

نصمم نفس التطبيقات السابقة ونكتب في حدث النقر التابع للزر الكود التالي:

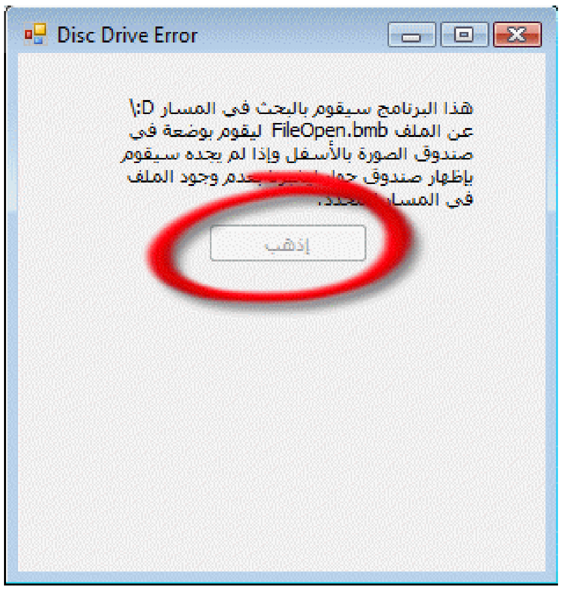
```
Try
         PictureBox1.Image = _
           System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
      Catch
```

```
        Retries += 1
        If Retries <= 2 Then
            MsgBox("تأكد من وجود الملف في المسار D:\")
        Else
            MsgBox("لا يمكنك المحاولة مرة أخرى")
            Button1.Enabled = False
        End If
    End Try
```

ونكتب هذا الكود قبل حدث النقر على الزر Click:

```
Dim Retries As Short
```

وبعد تنفيذ البرنامج ومحاولة النقر على الزر "اذهب" أكثر من مرتين يتم تعطيل إمكانية الضغط على الزر بسبب الكود Button1.Enabled = False وذلك بسبب تكرار العملية بدون نجاح ويكون الزر على الفورم غير فعال كما في هذه الصورة:



استخدام جُمَل Try...Catch المتداخلة

يمكنك استخدام جمل الـــ Try...Catch بشكل متداخل، بمعنى إضافة جملة Try...Catch بداخل جملة أخرى Try...Catch كما في المثال التالي:

```
Try
   PictureBox1.Image = _
     System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
   MsgBox("تأكد من وجود الملف في المسار المحدد ثم إضغط موافق")
   Try
      PictureBox1.Image = _
        System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
   Catch
      MsgBox("لا يمكنك المحاولة مرة أًخرى")
      Button1.Enabled = False
   End Try
End Try
```

ففي المثال السابق يقوم البرنامج بالسماح لنا بمعاودة الكرة مرة أَخرى ومحاولة تحميل الصورة من المسار المحدد وإذا لم يفلح فإنه يقوم بتطبيق الأمر البرمجي التالي لكلمة Catch الأولى وهو المحاولة مرة أَخرى أو تعطيل التعامل مع الزر Button1.

في الفصل العاشر من هذا الكتاب إن شاء الله ستتعلم كيف يتم تصميم دالة توجد فيها العديد من جمل الـــ Try...Catch المتداخلة.

مقارنة جُمَل صيد الأخطاء بخطوط الدفاع التقنية البرمجية

جُمَل صيد الأخطاء ليست الحل الوحيد للتعامل مع الأخطاء وإنما الحلول البرمجية أو خطوط الدفاع البرمجية قد تكون حل أمثل للتعامل مع الأخطاء فمثلاً بدل من صيد الخطأ (الملف غير موجود) يمكننا

استخدام الطريقة File.Exists التابعة للدوت نت فريم ورك تحت مجال الأسماء System.IO لمعرفة ما إذا كان الملف موجوداً أو ليس موجود. أنظر الكود التالي:

```
If File.Exists("d:\fileopen.bmp") Then
   PictureBox1.Image = _
     System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Else
   MsgBox("الملف غير موجود في المسار D:\")
End If
```

فبدلاً من صيد الخطأ (الملف غير موجود) تم الاستعانة بـــجملة If الشرطية بالإضافة إلى الطريقة File.Exists فإذا كان الملف موجود فسيقوم التطبيق بفتحه ووضعه في مربع الصورة على الفورم وإذا لم يوجد سيظهر لنا صندوق حوار يقول "الملف غير موجود في المسارD:\".

ملاحظة: بما أن الطريقة File.Exists تتبع مجال الأسماء System.IO في دوت نت فريم ورك فلا بد من استدعاء مجال الأسماء في منطقة الكود وذلك بوضع الكود التالي أعلى منطقة الكود:

```
Imports System.IO
```

وكذلك فبدلاً من الدخول في خطأ قد ينهار بسببه البرنامج بسبب القسمة على المقام الذي يساوي صفر، فبإمكاننا تصميم جملة شرطية تشترط إذا كان المقام يساوي صفر فإن التطبيق يرفض العملية ويوضح السبب بأن المقام يساوي صفر.

وبدلاً من الخلط بين الحروف والأرقام في تطبيق الآلة الحاسبة بإمكاننا منع المستخدم من إدخال غير الأرقام في مربع النص في الآلة الحاسبة لئلا يحدث خطأ عند محاولة البرنامج جمع أو ضرب الحروف.

السؤال الذي يتبادر إلى الذهن الآن متى نستخدم جُمَل صيد الأخطاء ومتى نستخدم خطوط الدفاع البرمجية في برامجنا: والإجابة هي يجب علينا استخدام مزيج من جُمَل صيد الأخطاء ومن خطوط الدفاع البرمجية في برامجنا بشكل علمي ومحسوب فمثلاً للأخطاء المتوقعة المعلومة نستخدم خطوط دفاع وللأخطاء المتوقعة الغير معروفه نستخدم جمل صيد الأخطاء. وكذلك إذا كان الأخطاء قليلة نستخدم خطوط الدفاع البرمجية وإذا كانت الأخطاء كثيرة نستخدم جمل صيد الأخطاء. وللعلم خطوط الدفاع البرمجية المبنية على أساس تقني منطقي أسرع في التجاوب من جُمَل صيد الأخطاء فالبرنامج الذي يبحث عن وجود الملف أو لا بأداة الشرط If والطريقة File.Exists أسرع من البرنامج الذي يستخدم جمل صيد الأخطاء لمعرفة وجود الملف من عدمه. وعليه فيفضل استخدام خطوط الدفاع البرمجية إلا في الحالات الموضحة أعلاه.

خطوة إضافية إلى الأمام، استخدام Exit...Try :

إذا راجعت الأكواد التي صممناها في هذا الفصل لصيد الأخطاء ستلاحظ بأننا نغلق جملة Try...Catch بالعبارة End Try والتي بدورها تقوم بإغلاق جملة صيد الأخطاء وينتقل البرنامج للجملة التي تليها في الكود ليقوم بتنفيذها (إذا وجدت).

لكن عند استخدامنا للجملة Exit Try في جملة صيد الأخطاء Try...Catch يقوم البرنامج مباشرة بمغادرة جملة صيد الأخطاء عند الوصول إليها (Exit Try) بدون مراجعة بقية الأكواد التي قد تكون بين ثنايا جملة صيد الأخطاء Try...Catch وحتى قبل أن يصل البرنامج إلى العبارة End Try.

لاحظ الكود التالي والمأخوذ من التطبيقات السابقة ولكننا أضفنا له عبارة Exit Try كتطوير لجملة صيد الأخطاء كالتالي:

```
Try
  If PictureBox1.Enabled = False Then Exit Try
  PictureBox1.Image = _
    System.Drawing.Bitmap.FromFile("d:\fileopen.bmp")
Catch
```

```
    Retries += 1
    If Retries <= 2 Then
        MsgBox("تأكد من وجود الملف في المسار D:\")
    Else
        MsgBox("لا يمكنك عرض الصورة")
        Button1.Enabled = False
    End If
End Try
```

لاحظ ثاني سطر في الكود والذي يتأكد من أن صندوق الصورة مفعل وجاهز لاستقبال الصورة فإذا كانت الخاصية لصندوق الصورة Enabled تساوي False فهذا يعني أن صندوق الصورة لا يمكنه استقبال الصورة ولا داعي لمحاولة إدراج صورة بداخله وعلية فإذا لم يكن صندوق الصورة مستعداً لقبول الصورة فالحل هو Exit Try بدون المضي قدماً في جملة صيد الأخطاء. أما إذا كان جاهزا لاستقبال الصورة وذلك بأن تكون الخاصية لصندوق الصورة Enabled تساوي True فسيقوم البرنامج بمواصلة جملة صيد الأخطاء، بسبب وجود أداة If الشرطية (أنظر السطر الثاني في الكود).

مبروك عليك هذه المعرفة التي اكتسبتها بقراءتك وتطبيقك حتى هذه النقطة من الكتاب ونستطيع القول بأنك قد فهمت العديد بل معظم أساسيات البرمجة ولكننا الآن سننحني منحنى آخر وذلك بالغوص عميقاً في بحر الدوت نت لنكتشف الدرر وذلك في الفصل القادم حيث سنتعلم كيف نبني الدوال والإجراءات البرمجية modules and procedures.

خلاصة الفصل التاسع:

من اجل أن	قم بالتالي
تحديد الأخطاء وقت	نقوم ببناء جملة لصيد الأخطاء باستخدام صيغة واحدة أو أكثر من

تشغيل البرنامج ومعالجتها	Try...Catch فمثلاً الكود التالي يحدد الأخطاء التي قد تواجهنا بسبب عدم وجود الملف أو بسبب الخطأ في كتابة المسار: Try PictureBox1.Image = _ System.Drawing.Bitmap.FromFile _ ("d:\fileopen.bmp") Catch MsgBox("تأكد من وجود الملف في المسار المحدد") Finally MsgBox("تم إنهاء العملية") End Try
لفحص خطأ محدد باستخدام جُمَل فحص الأخطاء	نستخدم الخاصية Err.Number لفحص خطأ محدد والتعامل معه، مثال: Try PictureBox1.Image = _ System.Drawing.Bitmap.FromFile _ ("d:\fileopen.bmp") Catch When Err.Number = 53 MsgBox("تأكد من وجود الملف في المسار المحدد") Catch When Err.Number = 7 MsgBox("تأكد من أن الملف المختار هو صورة", , _

Err.Description) Catch MsgBox("حدثت مشكلة وقت تحميل الملف", , _ Err.Description) End Try	
راجع الفصل لمعرفة سبب اختياري لكلمة عمداً، لتعمد إنشاء خطأ نستخدم الطريقة Err.Raise فمثلاً يمكننا اختلاق الخطأ (عفواً القرص ممتلئ)مثال: Try Err.Raise(61) Catch When Err.Number = 61 MsgBox("عفواً القرص ممتلئ") End Try	صناعة خطأ عمداً
نستطيع كتابة جمل صيد أخطاء Try…Catch بشكل متداخل كالتالي: Try PictureBox1.Image = _ System.Drawing.Bitmap.FromFile _ ("d:\fileopen.bmp") Catch MsgBox("تأكد من وجود الملف في المسار المحدد") Try	كتابة جُمَل صيد أخطاء Try…Catch متداخلة

PictureBox1.Image = _ System.Drawing.Bitmap.FromFile _ ("d:\fileopen.bmp") Catch MsgBox("تم إلغاء إمكانية عرض الصورة") Button1.Enabled = False End Try End Try	
نستخدم العبارة Exit Try للخروج من جملة صيد الأخطاء Try...Catch بدون مراجعة بقية الشروط بداخل الجملة، إذا استلزم الأمر، مثلاً إذا كان صندوق الصورة غير جاهز لاستقبال الصورة فبإمكاننا الخروج قبل تكملة Try...Catch، مثال: (راجع الفصل إذا كان المثال التالي غير واضح) If PictureBox1.Enabled = False Then Exit Try	الخروج من جملة صيد الأخطاء Try...Catch بدون المواصلة إلى النهاية

الفصل العاشر: كتابة دوال وإجراءات ووحدات وأحداث برمجية Creating Modules and Procedures

في الفصول التسعة السابقة تعلمنا كيف نضع كود معين أو نصمم كود معين ليعمل تحت حدث معين فمثلا كود لإغلاق البرنامج تحت حدث النقر على الزر Button1_Click أو تنفيذ أمر معين تحت حدث المؤقت Timer1_Tick أو تعبئة مربعات النص تحت حدث تحميل الفورم Form1_Load، وإذا لاحظت ستجد بأن جميع الأوامر يجب أن تكتب بين Private Sub و End Sub أو لا بد أن تكتب بداخل خانة معينة أو وحدة برمجية محددة أو بداخل حدث معين، وحتى تعريف المتغيرات إذا عرفنا متغير بداخل وحدة برمجية معينة مثل تعريف متغير Name على أنه String بداخل حدث Load التابع للفورم لن يكون المتغير معرفاً بداخل حدث Click التابع للزر Button وإنما لابد من تعريفه مرة أُخرى بداخل حدث Click.

وللتوضيح نقول أنه فقط المتغيرات العامة المعرفة بداخل كيانات Public أو Global هي التي نستطيع استخدامها في التطبيق بكامله، وعليه فاستخدام التعريف العام للمتغيرات يساعدنا كثيراً في اختصار الوقت وفي التقدم في الحلول المنطقية للأمور البرمجية.

سنتعرف في هذا الفصل على كيفية بناء وحدات منطقية Modules وهي عبارة عن وحدات برمجية في البرنامج تحتوي على متغيرات عامة ودوال برمجية وإجراءات وأحداث نستطيع استخدامها على طول البرنامج. سوف نتعلم كيفية تعريف واستخدام المتغيرات العامة وسوف نتعلم كيفية بناء إجراء عام يمكننا استخدامه في أكثر من تطبيق، وبالطبع سيسهل علينا عملية البرمجة وسيحفظ الوقت. المهارات التي سنتعلمها هامة جداً للمشاريع الكبيرة وكذلك وللعمل كفريق.

العمل مع الوحدات البرمجية الـــ Modules

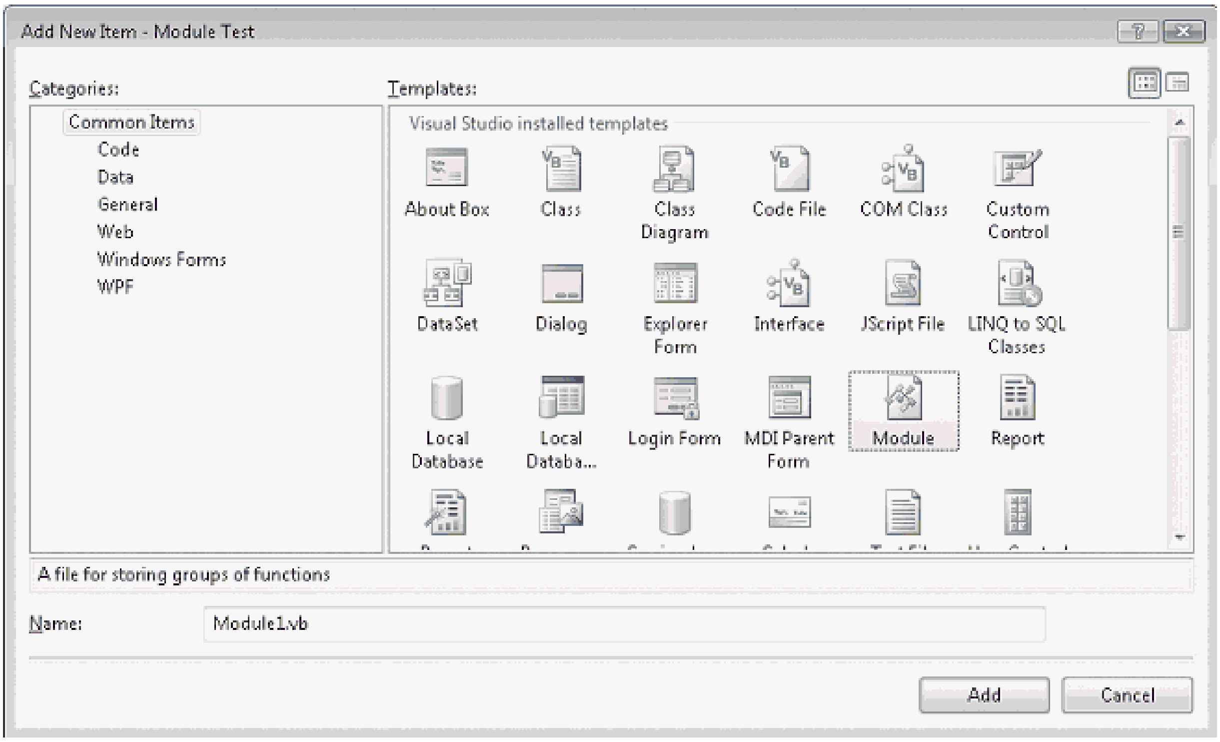
كلما ستقوم ببرمجة البرامج الكبيرة والمشاريع المتوسطة والعملاقة والتي تحتوي على العديد من النوافذ والإجراءات ستستخدم العديد من المتغيرات وبشكل متكرر، وبشكل أساسي في بيئة التطوير عندما نقوم بتعريف متغير ما فإن هذا المتغير يمكن استخدامه داخلياً فقط بداخل الحدث الذي تم تعريف المتغير فيه أو بداخل الإجراء البرمجي المحدد وإذا توسعت المسألة أكثر وعرفنا المتغير في أعلى منطقة الكود التابعة للفورم فيمكننا استخدام المتغير في داخل الكود لذلك الفورم الواحد فقط. إذا احتجنا لهذا المتغير في فورم آخر فيجب علينا تعريفه مرة أخرى وهذا لا يناسب المشاريع الكبيرة لذلك كان لابد لنا من استخدام الوحدات البرمجية المعروفة بـــ Modules والتي تساعدنا في اختصار الوقت وتوفر علينا تكرار تعريف المتغير في أكثر من فورم على مستوى المشروع أو التطبيق. فالـــ Module هي وحدة برمجية يمكننا تعريف المتغيرات والإجراءات والأحداث بداخلها واستخدامها في أي منطقة في المشروع. في فيجوال بيسك 6 كانت الـــ Module تنتهي بلاحقة bas. أما في الدوت نت فمعظم العناصر الداخلة في المشروع تنتهي بلاحقة vb. ومنها الـــ Module والنوافذ الـــ Forms وغيرها من مكونات المشروع.

التشابه بين النوافذ الـــ Forms والوحدات البرمجية الـــ Modules هو أنهما يتم فهرستهما في مستكشف المشروع Solution Explorer (أعلى يمين بيئة التطوير) ونستطيع النقر على نافذة معينة في مستكشف المشروع ومشاهدة تصميم النافذة والكود التابع لها لكن يوجد كود فقط للوحدات

البرمجية Modules ولا يوجد تصميم مثل النوافذ. يوجد تشابه بين الكلاس Class وبين الوحدات البرمجية الــ Modules في بعض الأمور ولكن تختلف الوحدات البرمجية الــ Modules عن الكلاس Class بأنها ليست كائنيه التوجه object-oriented ولا تحدد هيكل وخصائص الكائنات ولا يمكن وراثتها، إذا كانت هذه المصطلحات صعبة من وجهة نظرك فأعلم بأنك لست الوحيد في ذلك، ولا تخاف ففي الفصل السادس عشر سنتعرف جميعاً على هذه المصطلحات.

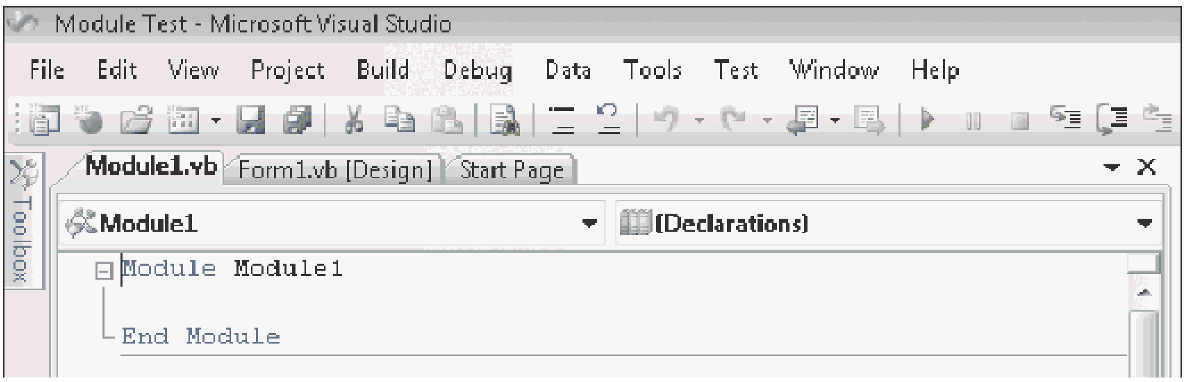
إنشاء وحدة برمجية Creating a Module

لإنشاء وحدة برمجية Module جديدة نذهب إلى مستكشف المشروع Solution Explorer (أعلى يمين بيئة التطوير) ونضغط بالماوس Right-Click على اسم المشروع ونختار Add ثم New Item ومن النافذة التي ستظهر نختار Module ونسميها بالاسم الذي نريد أو الموافقة على الاسم المقترح من قبل بيئة التطوير وهو Module1.vb، وحتى إذا وافقنا ثم فكرنا بتغيير الاسم لاحقاً نستطيع ذلك بالذهاب إلى مستكشف المشروع Solution Explorer ونختار الــ Module ونضغط بالماوس Right-Click ثم نختار Rename ونسميها بالاسم الذي نريد، أو حتى من نافذة الخصائص في الخاصية Name. على كل نقوم الآن بإنشاء وحدة برمجية Module جديدة فقط. انظر الصورة التي نختار منها الــ Module :



ستلاحظ في الصورة أعلاه العديد من المكونات التي يمكننا إضافتها للمشروع والتي توفرها بيئة التطوير من ضمنها نافذة دخول Login Form ونوافذ حوار Dialog وغيرها من النوافذ والمكونات التي سنتعرف عليها في الفصل السادس عشر.

بعد الضغط على الزر Add في الصورة أعلاه ستقوم بيئة التطوير بإضافة وحدة برمجية Module وفتحها كما في الصورة التالية:



الآن أي متغير ستقوم بتعريفة هنا يمكنك استخدامه في أي فورم على طول المشروع.

العمل مع المتغيرات العامة Working with Public Variables

تعريف المتغيرات العامة بداخل الوحدات البرمجية الـــ Modules سهل للغاية فقط قم بكتابة كلمة Public ثم اسم المتغير ثم As ثم نوع المتغير كما في المثال التالي الذي يقوم بتعريف المتغير Name كمتغير من النوع String :

```
Public Name As String
```

ما سيجعل الأمر أكثر متعه هو الإكمال التلقائي في بيئة التطوير. سنأخذ مثال على تعريف المتغير في وحدة برمجية Modules هل تتذكر برنامج الرقم المحظوظ برنامج Arqam الذي قمنا بتصميمه في بداية هذا الكتاب (موجود في المرفق رقم 002)، نعم سنذهب الآن إلى ذلك البرنامج لنضيف له وحده برمجية Module ونستفيد من المتغير بداخلها، سنضيف متغير باسم Wins ليقوم هذا المتغير بحساب كم عدد المرات التي ربحت فيها كلما دارت وتغيرت الأرقام (كلما ظهرت الصورة فهذا يعني بأنك ربحت، ومن أجل أن تظهر الصورة لا بد أن يظهر الرقم 7 في أحد الليبلات، هل تتذكر هذا). بعد فتح التطبيق Arqam نقوم بإضافة ليبل له ليظهر كما في الصورة:



ثم نضيف هذا الكود في الوحدة البرمجية الـــ Module قبل End Module:

```
Public Wins As Short
```

الكود أعلاه يقوم بتعريف المتغير Wins كمتغير عام في المشروع بالكامل، وبعدها نضيف هذا الكود في الحدث التابع للزر "ابدأ" في الفورم بعد PictureBox1.Visible = True:

```
If (Label1.Text = "7") Or (Label2.Text = "7") Or (Label3.Text = "7") _
Then
   Beep()
   Wins = Wins + 1
   Label5.Text = ("مرات الربح: " & Wins)
```

ثم نقوم بتشغيل البرنامج ونضغط على زر ابدأ أكثر من مرة. ماذا ستلاحظ؟

ستجد نسخة من المشروع أعلاه في المرفق رقم 029.

إنشاء إجراء Creating Procedures

الإجراء هو عبارة عن مجموعة من الأوامر البرمجية ذات العلاقة فيما بينها التي تقوم بمهمة معينة. وتنقسم إلى قسمين:

- إجراءات الدوال: وهي الإجراءات التي يتم استدعائها بواسطة اسمها من أي حدث، وعادة ما تستخدم للعمليات الحسابية وتستقبل مدخلات وترجع لنا قيم.
- الإجراءات Sub procedures وتستدعى بالاسم من أي حدث، وتستقبل مدخلات وترجع لنا مخرجات معدلة على شكل قائمة List، والفرق بينها وبين إجراءات الدوال بأنها لا ترجع لنا قيمة لها علاقة مع نوعها، وتستخدم عادة لاستقبال المدخلات وعرض المخرجات والخصائص.

كتابة إجراءات الدوال Writing Function Procedures:

إجراءات الدوال هي عبارة عن مجموعة من الأوامر البرمجية المكتوبة بين Function و End Function هذه الأوامر البرمجية تقوم بعمليات محددة حسابية أو منطقية أو غيرها وتخرج لنا مجموعة من المخرجات، بإمكانك استدعاء الدالة بواسطة كتابة اسمها في الكود مع وضع المعامِلات اللازمة. المعاملات هي البيانات التي تجعل الدالة تعمل ولابد من وضع هذه المعاملات

بين قوسين ويتم الفصل بين كل واحدة منها والأخرى بواسطة الفاصلة ، . ببساطة استخدام الدوال هو بالضبط مثل استخدام الدوال المرفقة أو الطُرق المرفقة ضمن بيئة التطوير مثل Int, Rnd، أو FromFile(لعلك تتذكر هذه الكلمات ويمكنك مراجعة الصفحات السابقة من هذا الكتاب).

ملاحظة: الدوال التي يتم تعريفها بداخل الحدات البرمجية الـــ Modules تعتبر عامة مباشرة لذلك تستطيع استخدامها في أي منطقة في الكود على طول المشروع.

بناء الدالة Function Syntax

البناء الأولي للدوال يأتي في مثل هذا الشكل:

```
Function FunctionName([arguments]) As Type
    function statements
    [Return value]
End Function
```

الكلمة FunctionName تعني اسم الدالة، Type نوع المخرجات من هذه الدالة في فيجوال بيسك 6 كان بإمكاننا عدم تحديد نوع المخرجات أما الآن فلا بد علينا من توضيح نوع المخرجات في فيجوال بيسك 2008، وإذا لم نحدد فإن بيئة التطوير تعتبر المخرجات من النوع Object.

Arguments هي المعاملات الاختيارية التي توجد في الدالة ويفصل بين كل معامل وآخر الفاصلة ، ولابد من تحديد نوع كل معامل على حدة وبشكل طبيعي تضيف بيئة التطوير الكلمة ByVal لكل معامل وهناك فرق بين ByVal و ByRef.

استدعاء دالة Calling a Function Procedure

لاستدعاء دالة ما في داخل حدث معين أو في أي منطقة من الكود نستدعيها كالمثال التالي الذي يستدعي دالة TotalTax (التي تحسب إجمالي الضريبة) وتضع الرقم الناتج في داخل مربع نص.

```
TextBox1.Text = TotalTax(500)
```

هذا الكود يستدعي الدالة TotalTax والتي بدورها تحسب إجمالي الضريبة المستحقة على المبلغ 500 دولار، ويتم وضع المخرجات أو النتيجة في صندوق النص التابع لــ TextBox1.

أما الرقم 500 فهو الرقم الذي نريد احتساب الضريبة له ويمكننا وضع متغير بدل عنه كما في المثال:

```
Dim X as Single
X= 500
TextBox1.Text = TotalTax(X)
```

استخدام الدوال في العمليات الحسابية:

باستخدام برنامج الرقم المحظوظ الذي صممناه في أول الكتاب وطورناه قبل قليل بحساب عدد مرات الربح، سنصمم الآن دالة تقوم بحساب نسبة الفوز بقسمة عدد مرات الفوز على عدد مرات ضغط الزر ابدأ. لنعد الآن إلى المشروع في المرفق 029 ونقوم بفتحه ونضيف ليبل إضافي للفورم ونعدل على خصائصه حتي يصير كما في الصورة التالية:



نذهب إلى الوحدة البرمجية الــ Module ونكتب الكود التالي فيها:

```
Public Spins As Short
Function HitRate(ByVal Hits As Short, ByVal Tries As Short) As String
   Dim Percent As Single
   Percent = Hits / Tries
   Return Format(Percent, "0.0%")
End Function
```

حيث تم تصميم دالة لحساب نسبة الربح بقسمة الــ Hits على Tries أو بقسمة عدد مرات النجاح على العدد الإجمالي للضغط على الزر ابدأ. ثم إخراج القيمة الناتجة على هيئة نسبة.

اسم الدالة أعلاه HitRate ونوعها نصي String لان نوع المخرجات نصية (لإن المخرجات النسبية عبارة عن مخرجات نصية).

نقوم الآن بفتح الفورم، منطقة الكود، وفي الحدث Click التابع للزر Button1 نقوم بتعديل الكود وذلك بإضافة الكود التالي بعد السطر الثالث (أي بعد السطر = Label3.Text (((CStr(Int(Rnd() * 10 نكتب الكود :

```
Spins = Spins + 1
```

والذي يحسب لنا عدد مرات النقر على الزر "ابدأ"، ثم نضيف كود استدعاء الدالة في أخر سطر في حدث النقر للزر Button1 وكالتالي:

```
Label6.Text = HitRate(Wins, Spins)
```

هل عرفت ما ستقوم به الدالة الآن، نعود الآن إلى الوحدة البرمجية الــ Module ونقرأها من جديد لنعرف بأننا سمينا الدالة HitRate ثم عرفنا لها اثنين من المعاملات وهما Hits و Tries على أنهما متغيرات رقمية وتم حساب القيمة الناتجة من قسمة Hits على Tries وإسنادها إلى متغير جديد سميناه Percent (متغير رقمي من النوع Single) فقط عند الحصول على

المخرجات من الدالة HitRate سنحصل عليها على هيئة نسبية بسبب السطر الأخير في الكود في الدالة "Return Format(Percent, "0.0%.

وعندما نستدعي الدالة في منطقة الكود في الفورم نطلب منها أن تقسم عدد مرات الفوز الـــ Wins على عدد مرات النقر على الزر "ابدأ" Spins ، وللتذكير فقط قد قمنا في المثال السابق بإسناد عدد مرات الفوز إلى المتغير Wins وقمنا الآن بإسناد عدد مرات الضغط على الزر "ابدأ" إلى المتغير Spins في الكود Spins = Spins + 1 .

نقوم الآن بتجربة التطبيق ثم حفظه وإغلاقه، نسخة من التطبيق موجودة في المرفقات برقم 030.

كتابة "إجراءات مصغرة" Sub Procedures

(للتوضيح مرة أُخرى فبعض المصطلحات لم استطع ترجمتها بصورة صحيحة لذلك قمت بوضع المصطلح الأصلي). الإجراءات تعتبر مشابهه للدوال ماعدا فرق واحد وهو أن الإجراءات لا ترجع لنا قيمة متناسقة مع نوعها، وتستخدم الإجراءات Sub procedure في الحصول على المدخلات من المستخدم وكذلك في عرض أو طباعة المعلومات، أو التعامل مع بعض الخصائص على حسب الحالة.

الصيغة العامة للإجراءات المصغرة Sub procedure

الصيغة العامة البسيطة للإجراءات المصغرة Sub procedure كالتالي:

```
Sub ProcedureName([arguments])
   procedure statements
End Sub
```

ProcedureName اسم الإجراء، arguments المعاملات ويفصل بين كل واحدة والأخرى فاصلة ، كل معامل لابد أن نحدد نوعه سيقوم فيجوال بيسك بتعريف المعامل على الأساس ByVal، procedure statements الجمل البرمجية.

إذا قمنا بتعريف الإجراءات في داخل الـــ Module فإنها تعتبر عامة ويمكن استدعاؤها من أي مكان في الكود.

تعريف المعاملات المضمنة في داخل الإجراءات يتم بطريقتين ByRef، ByVal ولكل طريقة ميزات مختلفة عن الأخرى وسنعرف الفروق بين الطريقتين في نهاية هذا الفصل.

استخدام الإجراءات للتعامل مع المدخلات:

الإجراءات تستخدم لجلب المدخلات من المستخدم خاصة إذا كانت هذه المدخلات تأتي من أكثر من مصدر ولكن بنفس الهيئة الـــ Format. ففي هذا المثال ستكون المدخلات عبارة عن أسماء الموظفين وسيتم إضافتها إلى صندوق نص متعدد الأسطر. الإجراء سيحفظ لنا الوقت فبدلاً من كتابة نفس الأوامر عند كل زر مرة أخرى سنقوم بتعريف الإجراء في Module ومن ثم التعامل معها عند كل صندوق نص.

مثال:

افتح الفيجوال بيسك 2008 وقم بإنشاء تطبيق جديد اسمه Text Box Sub، بعد إنشاء التطبيق نقوم بإضافة ثلاثة أزرار إلى الفورم وعدد اثنين ليبلات وكذلك اثنين صناديق نص. ليكون الفورم بالشكل التالي:



ثم نضيف Module للمشروع ونضيف للـــ Module هذا الكود قبل End Module:

```
Sub AddName(ByVal Team As String, ByRef ReturnString As String)
    Dim Prompt, Nm, WrapCharacter As String
    Prompt = "قم بإدخال " & Team & " موظف"
    Nm = InputBox(Prompt, "صندوق مدخلات")
    WrapCharacter = Chr(13) + Chr(10)
    ReturnString = Nm & WrapCharacter
End Sub
```

فهذا إجراء مصغر تمت إضافته في الـــ Module ليصبح إجراء عام ولا حاجة لإضافته عند كل فورم في المشروع، اسم الإجراء AddName، له اثنين من المعاملات وهما Team ، ReturnString تم تعريفهما على أنهما من النوع النصي String، لكن هناك فرق بينهما وهو أن المعامل Team يعتبر ByVal بينما المعامل ReturnString يعامل على الأساس ByRef.

وبسبب أن المتغير ReturnString يعتمد على الأساس ByRef فإن أي تغيير في هذا المعامل في الإجراء سينتقل هذا التغيير تبعاً عند عملية الاستدعاء. أما المعاملات المعتمدة على الأساس ByVal فيتم الاعتماد على القيمة الأولية بغض النظر عن التغييرات اللاحقة.

المتغيرات Prompt و Nm تم استخدامهم لعرض صندوق إدخال ليتم كتابة الاسم فيه.

أما المتغير WrapCharacter فسيقوم بالقفز بالاسم من سطر إلى آخر لكي يكون كل اسم في سطر مستقل وذلك باستخدام الكود (Chr(13) + Chr(10))، ويمكننا إنشاء سطر جديد في صناديق النص متعددة الأسطر دائماً باستخدام هذا الكود.

لاحظ هنا بأننا نريد استخدام الإجراء في حفظ اسم الموظف على هيئة معينة، في أكثر من قسم (قسم المبيعات وقسم التسويق) فإذا تم كتابة الكود تحت كل زر Button على حدة فسنتأخر فترة أطول، دعونا نفترض بأن البعض سيقبل ببعض التأخير ماذا إذا أردنا تحديث وتطوير الكود، سنقوم بكتابة التحديث تحت كل زر لكن في حالة تصميم الإجراءات ووضع الإجراء بداخل وحدة Module فسيكون التعديل أو التحديث سهلاً وأكثر فعالية.

هذا الإجراء يستخدم صندوق الإدخال ليسمح للمستخدم بإدخال اسم الموظف، على أساس اثنين من المعاملات Team والتي تعني نوع أسم القسم (قسم المبيعات وقسم التسويق) و ReturnString وهو المتغير الذي سوف يحتوي على اسم الموظف وبشكل مرتب بحيث يكون اسم كل موظف جديد في سطر جديد. وبسبب تعريف المعامل ReturnString بنوع ByRef فإن أي تغيير في قيمة هذا المعامل في الإجراء سينتقل بدورة إلى مرحلة استدعاء المعامل.

لنستعرض الفورم مرة أُخرى ونضغط Double-Click على الزر إضافة اسم تحت خانة المبيعات ونكتب هذا الكود والذي يستدعي الإجراء الذي قمنا بتصميمه سابقاً:

```
Dim SalesPosition As String = ""
AddName("المبيعات", SalesPosition)
TextBox1.Text = TextBox1.Text & SalesPosition
```

قمنا بتعريف متغير نصي SalesPosition ثم قمنا باستدعاء الإجراء AddName وحددنا المعاملين الاثنين له بداخل القوسين وهما 1- المبيعات ويساوي Team في الإجراء الذي تم تعريفه وكذلك المتغير النصي SalesPosition يساوي ReturnString ثم حددنا بأن خانة النص في صندوق النص TextBox1 تساوي النص الموجود بداخلها بالإضافة إلى النص المضاف عن طريق الإجراء AddName. ثم نفتح الفورم مرة أُخرى ونضغط Double-Click على الزر إضافة اسم تحت خانة التسويق ونكتب هذا الكود في حدث النقر على الزر:

```
Dim MktPosition As String = ""
AddName("المشتروات", MktPosition)
TextBox2.Text = TextBox2.Text & MktPosition
```

وهو نفس الكود للزر السابق مع فرق وحيد وهو تغيير اسم القيم في المتغير Team.

الآن نقوم بتشغيل البرنامج ونقوم بعملية إضافة الأسماء ومشاهدة التغييرات في صناديق النص، (ملاحظة: نسخة من المشروع أعلاه موجود في المرفقات برقم 031.)

خطوة إضافية إلى الأمام:

إضافة المعاملات بـــ ByVal أو ByRef

في المثال السابق ذكرنا أنه باستطاعتنا إضافة المتغير للإجراء بطريقتين وهما بالقيمة ByVal أو بالمرجع ByRef ولكل طريقة مواصفات معينة دعونا الآن لنتعرف على الفرق بينهما. القيمة الافتراضية هي ByVal حيث يتم تمرير المعامل بهذه الطريقة إذا لم نحدد نوعية تمرير المعامل.

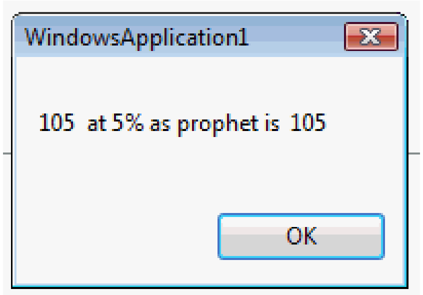
فإذا تم تمرير المعامل بالطريقة ByVal فسيتم تمرير قيمة المعامل بغض النظر عن التغيرات التي قد تحدث للمتغير، أما إذا تمت عملية تمرير المعامل بالطريقة ByRef فسيتم نقل المتغير بطريقة المرجع أي أنه إذا حدث تغيير في القيمة فسيتم نقل هذا التغيير إلى قيمة المتغير. دعونا نأخذ هذا الإجراء كمثال:

```
Sub CostPlusProphet(ByRef Cost As Single, ByRef Total As Single)
Cost = Cost * 1.05  ' أضف5 % إلى السعر
Total = Int(Cost)
End Sub

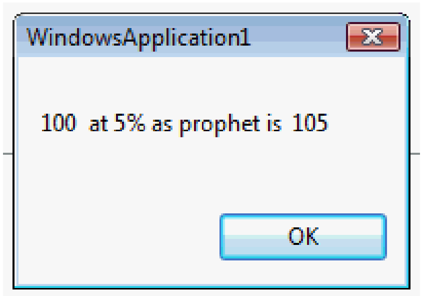
Dim Price, TotalPrice As Single
Price = 100
TotalPrice = 0
CostPlusProphet(Price, TotalPrice)
MsgBox(Price & "  at 5% as prophet is  " & TotalPrice)
```

لاحظوا أول سطر في الإجراء بأننا عرفنا التكلفة Cost بالطريقة ByRef وفي داخل الإجراء تم تغيير قيمة التكلفة Cost بإضافة 5% كربح أو كهامش ربحي وبما أن التكلفة الأولية تساوي 100 فإن قيمة المتغير Cost ستنتقل على أساس أنها تساوي 105 (أي بعد إضافة الـــ 5%) لأننا مررنا المتغير بالطريقة ByRef. لنجرب نفتح مشروع جديد ونضيف الكود من البداية حتى End

Sub في أي مكان داخل الفورم لكن لليس بداخل أي حدث (لإنه إجراء) ونضيف الكود الباقي تحت حدث النقر على زر أو تحت حدث تحميل الفورم وسنلاحظ هذا الصندوق الحواري:



أما إذا عدلنا الكود من ByRef Cost As Single إلى ByVal Cost As Single فسيظهر هذا:



والسبب في ذلك يعود إلى ما تم توضيحه سابقاً بأن عملية تعريف المتغير بالطريقة ByRef يتم نقل قيمة المتغير مع ملاحظة التغيرات التي قد تطرأ على المتغير أما تعريف المتغير بالطريقة ByVal فلا تهتم بالتغيرات التي تحدث على المتغير ويتم نقل قيمة المتغير الأولية بدون مراجعة المتغيرات. إذا عدنا إلى المثال أعلاه سنلاحظ بأن التغيير الذي حدث في المتغير هو تغيير في قيمة المتغير بسبب الأوامر البرمجية التي كتبناها في الإجراء وعلية فننتبه للأوامر التي قد نضيفها في الأكواد فإذا كان لهذه الأوامر تغيير في قيمة المتغير فلا ننسى أن نمرر المتغير بالطريقة ByRef أما إذا يوجد أي تغيير فلا بأس من استخدام الطريقة الافتراضية ByVal. ولا ننسى بأن لكل طريقة مميزات وعيوب حسب طبيعة البرنامج. وللتذكير فإن الطريقة الافتراضية هي ByVal بحيث إذا لم يتم تحديد طريقة تمرير المتغير فإن بيئة التطوير تعتمد هذه الطريقة في التمرير.

باختصار شديد لعملية تمرير قيمة المتغير:

- نستخدم ByRef إذا أردنا متابعة أية تغيير قد يحدث في قيمة المتغير بعد استدعاء المتغير من إجراء أو دالة معينة.
- نستخدم ByVal إذا أردنا كتابة قيمة المتغير الأولية بدون الانتباه إلى التغييرات التي قد تحدث بعد عملية الاستدعاء.
- نستخدم ByVal إذا لم نحدد طريقة معينة لتمرير قيمة المتغير.

خلاصة الفصل العاشر

من اجل أن	**قم بالتالي**
إنشاء وحدة برمجية Module	اختر Add New Item في شريط الأدوات ثم اختر Module وأضفها إلى مشروعك. أو من القائمة Project اختر Add Module أو اختر Add New Item ثم اختر Module.
تغيير اسم الـــ Module	قم باختيار الـــ Module في مستكشف المشروع وفي نافذة الخصائص وفي الخاصية Name قم بكتابة الإسم الجديد للـــ Module. أو اختر الـــ Module في مستكشف المشروع ثم انقر Right-Click واختر Rename ثم قم بكتابة الاسم الجديد.
حذف الـــ Module من البرنامج	انقر بالماوس Right-Click فوق الـــ Module في مستكشف المشروع ثم اختر Exclude from Project.
إضافة Module موجودة مسبقاً إلى مشروعنا	من قائمة Project اختر Add Existing Item ثم اختر الـــ Module الذي تريد إضافته ثم قم بإضافته للمشروع.
تعريف المتغيرات	المتغيرات العامة هي المتغيرات التي نحتاجها علة طول المشروع فإذا

العامة (المتغيرات الموجودة بطول المشروع)	اعتبرنا بأن سنة إنشاء الشركة متغير زمني فقد يكون هذا المتغير هام جداً لنا في أية نقطة من مراحل بناء المشروع ولذلك لابد من تعريفه مرة واحدة في المشروع بدلاً من إضاعة الوقت وتعريفة عند كل إجراء على حدة، ولتعريف المتغيرات العامة بداخل الوحدات البرمجية الـــ Modules بين السطرين Module و End Module باستخدام الكلمة Public كالتالي: Public TotalSale As Integer
إنشاء دالة عامة	الدالة العامة هي الدالة التي يمكن استخدامها في أي منطقة برمجية على طول المشروع ولإنشائها لابد من كتابتها بداخل Module بين Module و End Module ونكتب جميع أوامر الدالة بين Function و End Function كالتالي: Function HitRate(ByVal Hits As Short, ByVal _ Tries As Short) As String Dim Percent As Single Percent = Hits / Tries Return Format(Percent, “0.0%”) End Function وللعلم الدوال المعرفة بداخل الوحدات البرمجية الـــ Module تعتبر دوال عامة بشكل افتراضي بدون الحاجة إلى كتابة كلمة Public.
استدعاء إجراء من دالة معينة	لاستدعاء إجراء معين موجود في دالة محددة نكتب اسم الدالة متبوعاً بالمعاملات المتوفرة للدالة بداخل قوسين ونقوم بإسناد القيمة المتوقعة إلى خاصية معينة مثل خاصية النص في صناديق النص كالتالي:

	lblRate.Text = HitRate(Wins, Spins) ففي المثال أعلاه قمنا باستدعاء الدالة HitRate وكتابة معاملاتها بداخل القوسين ومن ثم إسناد القيمة الناتجة إلى خاصية النص في صندوق النص lblRate.
إنشاء إجراء عام	نقوم بتعريف هذا الإجراء بداخل وحدة برمجية Module ونضع جميع الأوامر البرمجية التابعة للإجراء بين Sub و End Sub . الإجراءات تعتبر عامة بمجرد تعريفها في الـــ Modules. مثال للإجراء : Sub CostPlusInterest(ByVal Cost As Single, ByRef Total As Single) Cost = Cost * 1.05 Total = Int(Cost) End Sub فالإجراء CostPlusInterest يعتبر إجراء عام بمجرد تعريفة في Module.
استدعاء إجراء	نقوم بكتابة اسم الإجراء متبوعاً بالمعاملات الخاصة بالإجراء بين قوسين. مثال (استدعاء الإجراء CostPlusInterest): CostPlusInterest(Price, TotalPrice)
تمرير المعامل بالقيمة ByVal	نستخدم الكلمة ByVal عند تعريف المتغير في الإجراء أو الدالة كالمثال: Sub GreetPerson(ByVal Name As String)
تمرير المعامل بالمرجع	نستخدم الكلمة ByRef عند تعريف المتغير في الإجراء أو الدالة كالمثال:

ByRef	Sub GreetPerson(ByRef Name As String)

الفصل الحادي عشر: استخدام المصفوفات للتعامل مع البيانات الرقمية والنصية.

كما تعلمنا في الفصول الدراسية في المرحلة الثانوية (لمن يتذكر) المصفوفات بأنها الطريقة الأمثل للتعامل مع كمية كبيره من البيانات المتشابهة. أما برمجيا فنعلم بأن البرامج تعتمد بشكل أساسي على البيانات والمدخلات وتقوم البرامج بالتعامل مع هذه البيانات وتحليلها بحسب البرمجة المسبقة لهذه البرامج. وعند التعامل مع البيانات في فيجوال بيسك تعلمنا كيف نقوم باستخدام المتغيرات لحفظ هذه البيانات في متغيرات ثم تحويل أو إسناد قيمة هذه المتغيرات إلى خاصية معينة في صندوق نص أو غيره، لكن عندما يكون لدينا العديد (الكثير) من البيانات المتشابهة، فلابد لنا من استخدام المصفوفات للتعامل مع هذه البيانات وللعلم توجد المصفوفات في معظم أو في كل لغات البرمجة للتعامل مع البيانات فإذا كان لدينا مصفوفة فيها ثلاثة أعمدة وثلاثة صفوف فنستطيع استخدامها للتعامل مع تسعة من البيانات المتشابهة.

إنشاء المصفوفات:

لابد هنا أن نعلم بأنه عند كتابة المصفوفة في مكان ما فإن هذا المكان يحدد لنا نطاق استعمال المصفوفة، فمثلاً عند كتابة المصفوفة في داخل إجراء أو حدث معين فإننا نستخدم المصفوفة في داخل ذلك الإجراء فقط، أما إذا قمنا بكتابة المصفوفة في بداية الفورم (أعلى منطقة الكود في الفورم) فإن المصفوفة يمكن استخدامها على طول الكود في الفورم، وفي حالة كتابة المصفوفة بداخل وحدة برمجية Module فنستطيع استخدام المصفوفة في أي مكان في المشروع. وعند إنشاء أي مصفوفة فأنت باختصار تجيب عن هذه أو توضح المعلومات التالية عند تعريف المصفوفة:

- **اسم المصفوفة:** اسم المصفوفة هو الاسم الذي ستستخدمه لاستدعاء مصفوفتك في أي مكان في الكود واسم المصفوفة لابد أن يخضع لنفس الشروط اللازمة لاسم المتغير. راجع شروط كتابة اسم للمتغير والتي تعلمناها في الفصل الخامس.

- **نوع البيانات:** لابد لنا من تحديد نوعية البيانات التي سوف يتم تسجيلها في المصفوفة وفي أغلب الحالات تكون هذه البيانات من نوع واحد (كلها نصية أو كلها وقتية أو كلها رقمية) ونادراً ما تكون أكثر من نوع. لابد لك من تحديد نوعية البيانات داخل المصفوفة لتحصل على التعامل الأمثل للبيانات من قبل بيئة التطوير أما إذا لم تعرف نوعية البيانات التي سوف تستخدمها بداخل المصفوفة أو كان هناك أكثر من نوع من أنواع البيانات فلابد لك إذا من استخدام النوع Object للتعبير عن البيانات.

- **أبعاد المصفوفة:** لابد لك من تحديد مقاسات أو أبعاد المصفوفة فهناك مصفوفات ذات بعد واحد (مثل قائمة من البيانات) وهناك ذات بعدين (مثل جدول من البيانات) وهناك مصفوفات بثلاثة أبعاد للمصفوفات والتي تعالج مجموعة كبيرة ومعقدة ومتداخلة من البيانات.

- **عناصر المصفوفة:** عناصر المصفوفة أو عدد عناصر المصفوفة هو عدد العناصر التي ستتعامل معها بداخل المصفوفة وهذه العناصر تطابق عدد العناصر في القائمة التي ترتبها المصفوفة ولابد أن يبدأ الترتيب من الرقم 0 في فيجوال بيسك 2008. فإذا كان عدد العناصر تسعة فإن بيئة التطوير ترتبهم في قائمة من 0 إلى 8، حيث يأخذ العنصر الأول الترتيب صفر والعنصر الثاني الترتيب واحد وهكذا.

ملاحظة: المصفوفة التي تحتوي على عدد ثابت ومحدد من العناصر تسمى المصفوفة الثابتة Fixed-Size array والمصفوفة التي تحتوي على عدد متغير من العناصر كتلك التي يتغير عدد عناصرها في وقت تشغيل البرنامج بسبب البيانات التي تتعامل معها تسمى المصفوفة المتغيرة أو المصفوفة التفاعلية Dynamic array.

تعريف مصفوفة ثابتة Fixed Size array

القاعدة العامة لتعريف المصفوفات الثابتة كالتالي:

```
Dim ArrayName(Dim1Index, Dim2Index, ...) As DataType
```

Dim الكلمة التي تستخدم لتعريف المصفوفة ونستخدم كلمة Public بدلاً عنها إذا كانت المصفوفة بداخل وحدة برمجية Module. ArrayName اسم المصفوفة التي نريد تعريفها، Dim1Index عدد عناصر المصفوفة في البعد الأول في المصفوفة ويساوي نفس عدد عناصر المصفوفة ناقصاً منه 1 (لإن تسلسل المصفوفات يبدأ من صفر وليس من واحد)، أما Dim2Index فهو عناصر المصفوفة في البعد الثاني في المصفوفة ويساوي نفس عدد عناصر المصفوفة في البعد الثاني ناقصاً منه واحد. ونستطيع إضافة أبعاد جديدة بفصل كل بعد عن الآخر بفاصلة. DataType نوعية بيانات المصفوفة (نصية، وقتية، رقمية، أو أخرى).

لنفرض أن لدينا عشرة موظفين ونريد أن نعرف مصفوفة لتحتوى هؤلاء الموظفين فإن طريقة تعريف المصفوفة سيكون كالتالي: سنفترض بأن اسم المصفوفة Employees وبما أن عدد الموظفين عشرة فسيكون Dim1Index عدده تسعة (10–1). سيكون الكود كالتالي:

```
Dim Employees(9) As String
```

أما إذا عرفنا المصفوفة بداخل وحدة برمجية Module فسيكون التعريف كالتالي:

```
Public Employees(9) As String
```

نستطيع كتابة نفس التعريف السابق بطريقة أخرى كالتالي:

```
Dim Employees(0 To 9) As String
```

بحيث نذكر بداية ونهاية المصفوفة وبما أن عدد الموظفين عشرة ولابد أن تبدأ المصفوفة من صفر فسيكون التعريف كما هو مبين أعلاه. ملاحظة بأن هذه الطريقة من التعريف ليست مدعومة في فيجوال بيسك 2002 و 2003 وهي مدعومة فقط في نسختي 2005 و 2008. وللتنبيه هنا لمبرمجين فيجوال بيسك 6 بأننا الخيار الذي كان يسمح لك بجعل المصفوفة تبدأ من 1 أصبح غير مدعوم حالياً، بعبارة أخرى لابد أن يبدأ ترتيب عناصر المصفوفة من صفر وليس من واحد.

حجز الذاكرة لمصفوفة معينة:

عند تعريف مصفوفة يقوم الكمبيوتر عند تشغيل البرنامج الذي يحتوي على المصفوفة بحجز مكان لهذه المصفوفة فمصفوفة الموظفين الموضحة أعلاه يقوم الكمبيوتر بحجز ذاكرة لها كالتالي:



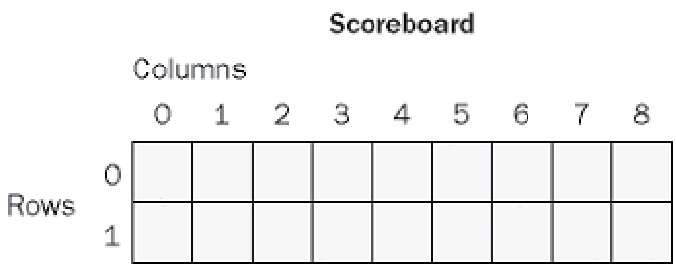
لاحظ كيف يبدأ الكمبيوتر بترتيب الموظفين العشرة من صفر إلى تسعة. هذا في المصفوفة ذات البعد الواحد فكيف تتم العملية إذا كان لدينا مصفوفة ذات بعدين ولنفرض أننا نريد مصفوفة تحتوي على جدول لنتيجة مباراة كرة السلة وكما نعرف أن عدد مراحل المباراة تسع ويوجد لدينا فريقين فلابد من تصميم مصفوفة تحتوي على صفين وتسعة أعمدة فنكتب الأمر التالي:

```
Dim Scoreboard(1, 8) As Short
```

جميل أن نتذكر بأن المصفوفة تبدأ من الصفر، بحيث يمكننا استخدام الأمر التالي بدلاً عن الأول:

```
Dim Scoreboard(0 To 1, 0 To 8) As Short
```

الأمر أعلاه سينتج لنا هذه المصفوفة:



حيث لدينا صفين (الصف 0 والصف 1، ولدينا تسعة أعمدة).

التعامل مع عناصر المصفوفات:

للتعامل مع عناصر المجموعات سواءاً أردنا الحصول على قيمة معينة تابعة لعنصر معين في المصفوفة أو تعبئة عنصر معين في المصفوفة بقيمة معينة فلا بد لنا من تحديد ترتيب العنصر في المصفوفة ثم كتابة قيمته ليتم تعبئتها مثلا المصفوفة التابعة للموظفين والتي استخدمناها سابقاً لإضافة اسم للموظف السادس ولنفرض بأن اسم الموظف Leslie نكتب الكود التالي:

```
Employees(5) = "Leslie"
```

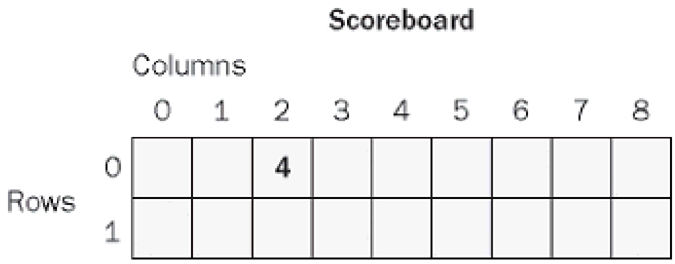
فسيتم تعبئة المصفوفة بذلك الاسم كالتالي:



وماذا سيحدث إذا أردنا تعبئة بيانات مصفوفة ذات بعدين مثل مصفوفة Scoreboard التي عرفناها سابقاً، بالطبع نستخدم نفس الطريقة لتعبئة عناصر المصفوفة بالقيم أو لسحب قيمة عنصر معين، لنفرض بأننا نريد تعبئة العنصر الثالث في الصف الأول بالقيمة 4، نكتب الكود التالي:

```
Scoreboard(0, 2) = 4
```

فسيتم تعبئة المصفوفة كالتالي:



تستطيع استخدام هذه الطريقة في تعبئة القيم أو في الحصول على قيم من داخل عناصر المصفوفة.

تصميم مصفوفة لخزن درجة الحرارة لأيام الأسبوع:

سنقوم الآن بتصميم مصفوفة ثابتة ذات بعد واحد تقوم بخزن درجة الحرارة لأيام الأسبوع وسنقوم بتعبئتها باستخدام صندوق الإدخال InputBox والتعامل معها باستخدام For Next Loop التي تعلمناها في الفصل السابع من هذا الكتاب. نستخدم طريقة For Next Loop للتعامل مع كل بند من بنود المصفوفة على حده إذا استلزم الأمر. ونقوم بعرض عناصر المصفوفة بداخل صندوق نصي بداخل الفورم. وفي نفس الوقت سنقوم بحساب متوسط حرارة الاسبوع.

بداية ونهاية المصفوفة: Lbound and Ubound

نستطيع تحديد تسلسل أعلى قيمة في المصفوفة وتسلسل أقل قيمة في المصفوفة بواسطة الدوال Ubound و Lbound، حيث هذه الدوال كانت متوفرة في النسخ القديمة من فيجوال بيسك حيث Lbound ترمز لتسلسل أقل قيمة في المصفوفة، وكما هو معروف بأن المصفوفات في فيجوال بيسك 2008 تبدأ من التسلسل صفر فإن الدالة Lbound سترجع لنا القيمة صفر، أما Ubound فترجع لنا تسلسل أعلى قيمة في المصفوفة فإذا كانت المصفوفة ذات عشرة عناصر فإن الدالة Ubound سترجع القيمة 9 لنا لإن التسلسل يبدأ من الرقم 0 كما ذكرنا سابقاً.

الآن نتابع تصميم مصفوفة لخزن درجة حرارة أيام الأسبوع:

١. نقوم بفتح بيئة التطوير ونتشيء مشروع جديد باسم My Fixed Array.

٢. نضيف صندوق نصي إلى الفورم.

٣. نعدل الخاصية MultiLine التابعة لصندوق النص إلى True لتسمح لنا بكتابة العديد من الأسطر بداخل صندوق النص.

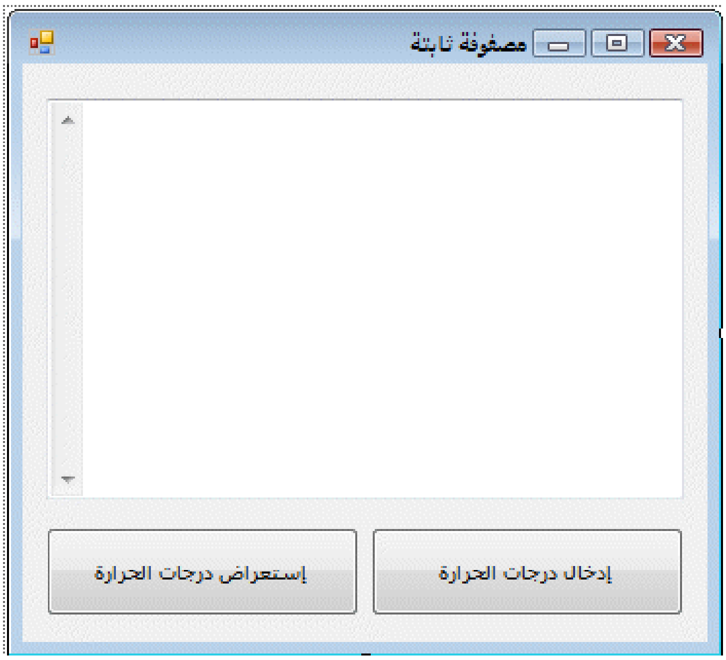
٤. نقوم بتعديل صندوق النص وتكبيره ليغطي معظم مساحة الفورم.

٥. نضيف إثنين أزرار إلى الفورم تحت صندوق النص.

٦. نعدل بعض الخصائص للمكونات على الفورم كالتالي:

TextBox1	Scrollbars	Vertical
Button1	Text	"إدخال درجات الحرارة"
Button2	Text	"إستعراض درجات الحرارة"
Form1	Text	"مصفوفة ثابتة"

ستكون الفورم كما في الشكل التالي:



٧. نقوم بتعريف المصفوفة في محرر الكود ليتم التعامل معها على طول الإجراءات في الفورم، لذلك نضع التعريف في بداية الفورم وتحت Public Class Form1 مباشرة،

وسيكون التعريف كالتالي:

```
Dim Temperatures(0 To 6) As Single
```

الكود أعلاه يعرف لنا مصفوفة من سبعه عناصر (من الصفر إلى الستة) ويحدد نوع عناصرها بالنوع Single ومنعاه رقم حقيقي.

٨. نفتح الفورم ثم نقوم بالضغط Double-Click على الزر "إدخال درجات الحرارة" ونكتب الكود التالي في الإجراء Click الخاص بالزر:

```
Dim Prompt, Title As String
Dim i As Short
Prompt = "قم بإدخال أعلى درجة حرارة لليوم."
For i = 0 To UBound(Temperatures)
    Title = "اليوم " & (i + 1)
    Temperatures(i) = InputBox(Prompt, Title)
Next
```

٩. نلاحظ أننا في الكود أعلاه استخدمنا التعبير For...Next لنبدأ من الصفر إلى أعلى تسلسل UBound في المصفوفة Temperatures. فكان باستطاعتنا كتابة الكود :

```
For i = 0 To 6
```

بدلاً من السطر الرابع من الكود. إذا كانت المصفوفة قد تكبر وتزيد عناصرها أو قد تصغر وتقل عناصرها فمن الأفضل استخدام UBound لمعرفة آخر تسلسل في المصفوفة.

نفتح الفورم ثم نقوم بالضغط Double-Click على الزر "استعراض درجات الحرارة" ونكتب الكود التالي في حدث Click التابع للزر:

```
Dim Result As String
Dim i As Short
Dim Total As Single = 0
```

```
Result = "أعلى درجات الحرارة خلال الأسبوع:" & vbCrLf & vbCrLf
For i = 0 To UBound(Temperatures)
    Result = Result & "اليوم " & (i + 1) & vbTab & _
      Temperatures(i) & vbCrLf
    Total = Total + Temperatures(i)
Next
Result = Result & vbCrLf & _
  "متوسط درجة الحرارة: " & Format(Total / 7, "0.0")
TextBox1.Text = Result
```

الكود أعلاه يستخدم الحلقة التكرارية For I = 0 To UBound(Temperatures) وتبدأ هذه الحلقة من صفر إلى أعلى قيمة في المصفوفة ليأخذ درجة الحرارة المخزنة في المصفوفة عن كل يوم ووضعها في صندوق النص TextBox1، طبعاً بعد كل درجة حرارة عن كل يوم سيضعها في صندوق النص كتبنا vbTab وهذا يجعل البرنامج يقفز القفزة Tab الموجودة في الكيبورد أما vbCrLf فينقلنا إلى السطر التالي.

يقوم البرنامج بحساب متوسك درجة الحرارة ويضعها في سطر جديد بواسطة الكود:

```
"متوسط درجة الحرارة: " & Format(Total / 7, "0.0")
```

الذي يقوم بقسمة إجمالي درجات الحرارة للأسبوع على الرقم ٧. ويضعها على هيئة "0.0" أي رقم وبعده فاصلة، نستطيع زيادة الأصفار بعد الفاصلة لنحصل على أرقام دقيقة لكننا لا نحتاج لمثل هذا الإحصاءات الدقيقة لإننا نحتاج لمعرفة درجة الحرارة، لكن إذا كان لدينا أرقام حساسة للكسور بعد الفاصلة فيمكننا تكثير الأصفار بعد الفاصلة.

ملاحظة: المثال موجود بالمرفقات بالرقم ٠٣٢.

تعريف مصفوفة تفاعلية Fixed Dynamic array

تعرفنا في المثال السابق على تصميم المصفوفة الثابتة (المصفوفة المعلوم عدد عناصرها) وتعلمنا كيف نستفيد منها في التعامل مع كمية كبيرة من البيانات أو مع قائمة من البيانات، خاصة عند استخدام الحلقات التكرارية معها، في الحقيقة قد لا نعرف عدد العناصر التي سنتعامل معها خلال فترة عمل التطبيق أو البرنامج (Runtime) في مثل هذه الحالة لا يمكننا استخدام المصفوفات الثابتة لإننا لا نعرف عدد عناصر المصفوفة وعليه فقد وفرت لنا بيئة التطوير نوع من المصفوفات وهي المصفوفات المتغيرة أو المصفوفات التفاعلية. هذا النوع من المصفوفات لا يحتاج لتحديد عدد عناصر المصفوفة في مرحلة الكود وإنما في مرحلة تشغيل البرنامج **Runtime**.

ولتعريف مصفوفة تفاعلية نقوم بتحديد أسم ونوع المصفوفة في مرحلة الكود وبدون ذكر عدد العناصر للمصفوفة، فمثلاً إذا كان لدينا مصفوفة لدرجات الحرارة باسم **Temperatures** نقوم بتعريفها كالتالي:

```
Dim Temperatures() As Single
```

بدون ذكر عدد عناصرها في مرحلة الكود. بعد كود تعريف المصفوفة نقوم بكتابة كود آخر يسمح للمستخدم بكتابة عدد عناصر المصفوفة في مرحلة تشغيل البرنامج **Runtime**، بطريقة أو بأخرى يمكننا السماح للمستخدم بكتابة عدد عناصر المصفوفة باستخدام الكود بدون كتابة عدد العناصر في مرحلة الكود وإنما إسناد العدد إلى متغير معين ويقوم مستخدم البرنامج بتعبئة هذا المتغير كالتالي:

```
Dim Days As Short
Days = InputBox("كم عدد الأيام؟", "قم بإنشاء المصفوفة")
```

المتغير لأيام المصفوفة هنا هو **Days** وسيقوم المستخدم بتعبئة قيمة المتغير بكتابة عدد الأيام بداخل صندوق الإدخال الذي سيظهر. بعد هذا سنقوم بإعادة تعريف المصفوفة (إنتبه لكلمة إعادة) لنحدد عدد عناصرها بحسب ما أدخل المستخدم كالتالي:

```
ReDim Temperatures(Days - 1)
```

الكلمة **ReDim** معناها أعد تعريف، ولكن قد تتساءل لماذا عرفنا المصفوفة بأقل من عدد الأيام المدخلة من قبل المستخدم برقم واحد حيث كتبنا في الكود **Days - 1** التفسير لهذه النقطة يعيدنا إلى تعريف المصفوفات الذي درسنا عنه سابقاً حيث أن كل مصفوفة يجب أن تبدأ من الصفر

لنفرض بأن عدد العناصر ٧ فإن المصفوفة ستبدأ بالصفر وستنتهي بالرقم ٦ ليكون عدد العناصر سبعة، أعد قراءة الجملة مرة ثانية لتعرف أكثر.

ملاحظة هامة: إذا كنا قد عرفنا المصفوفة وذكرنا عدد عناصرها فلا يمكننا إستخدام ReDim لتغيير عدد العناصر وإنما نستخدم ReDim فقط مع المصفوفة غير معلومة عدد العناصر فقط.

وللفائدة عند التعامل مع المصفوفات التفاعلية نستخدم UBound للتعامل مع آخر تسلسل في المصفوفة وخاصة في الحلقات التكرارية، لإننا لا نعرف عدد عناصر المصفوفة. مثال:

```
For i = 0 to UBound(Temperatures)
Temperatures(i) = InputBox(Prompt, Title)
Next
```

مثال على تصميم المصفوفات التفاعلية:

في هذا المثال سنقوم بالتعديل على المثال الذي تعلمناه سابقاً في تصميم المصفوفات الثابتة، لنعدل المصفوفة إلى مصفوفة تفاعلية كالتالي: نفتح المثال السابق الموجود في المرفقات برقم ٠٣٢ ونقوم بالتالي:

١- نقوم بفتح المشروع ونذهب إلى منطقة الكود وفي أعلى منطقة الكود بالذات نحذف عدد عناصر المصفوفة الذى كتبناه وهو : ٠ To ٦ وبهذا تتحول المصفوفة مباشرة إلى مصفوفة تفاعلية (متغيرة) وستكون الجملة البرمجية كالتالي:

```
Dim Temperatures() As Single
```

٢- نقوم بتعريف المتغير Days ليقوم بحفظ عدد الأيام (عدد عناصر المصفوفة) ونقوم بكتابة التعريف تحت جملة تعريف المصفوفة السابقة كالتالي:

```
Dim Days As Integer
```

٣- ننزل قليلاً بالماو إلى الحدث Click التابع للزر Button1 ونضيف الكود التالي (الكود المضلل باللون الأصفر) كالتالي:

```
Dim Prompt, Title As String
Dim i As Short
Prompt = "قم بإدخال أعلى درجة حرارة لليوم."
Days = InputBox("كم عدد الأيام؟", "قم بإنشاء المصفوفة")
If Days > 0 Then ReDim Temperatures(Days - 1)
For i = 0 To UBound(Temperatures)
   Title = "اليوم " & (i + 1)
   Temperatures(i) = InputBox(Prompt, Title)
Next
```

السطر الرابع في الكود أعلاه يظهر للمستخدم صندوق إدخال ليسمح له بكتابة عدد الأيام (عدد عناصر المصفوفة) أما السطر الخامس فيقوم بالتأكد من أن العدد المُدخل أكبر من صفر لإن تعريف المصفوفة التي عدد عناصر أقل من صفر يتسبب في ظهور خطأ. ولأننا سنقوم بطرح ١ من العدد فلابد أن يكون الرقم المُدخل أكبر من صفر. للتذكير فقط لا نحتاج لاستخدام UBound لإنها مُستخدمه من سابق.

٤- نذهب في الكود إلى الحدث Click التابع للزر Button2 وقم بتعديل الكود حتى يظهر كما الكود التالي: (الكود باللون الأصفر هو الكود المعدل)

```
Dim Result As String
Dim i As Short
Dim Total As Single = 0
Result = "أعلى درجات حرارة:" & vbCrLf & vbCrLf
For i = 0 To UBound(Temperatures)
    Result = Result & "اليوم " & (i + 1) & vbTab & _
     Temperatures(i) & vbCrLf
```

```
    Total = Total + Temperatures(i)
Next
Result = Result & vbCrLf & _
  "متوسط درجة الحرارة: " & Format(Total / Days, "0.0")
TextBox1.Text = Result
```

تم تعديل الكود أعلاه ليناسب المصفوفة التفاعلية، وتم تغيير النص الذي يظهر وحذف كلمة الأسبوع منه لإننا لا نعرف هل عدد الأيام سبع أو غير ذلك وكذلك تم تغيير طريقة الحصول على المتوسط فبدلاً من القسمة على الرقم ٧ حل المتغير Days بدل الرقم لتتم القسمة على عدد الأيام، وهذا مناسب بسبب أننا لا نعرف عدد الأيام.

٥- نفتح نافذة تصميم الفورم ثم نغير الخاصية Text للفورم من "مصفوفة ثابتة" إلى "مصفوفة تفاعلية"

٦- الآن انتهينا من تصميم تطبيق المصفوفة التفاعلية نقوم الآن بحفظ التطبيق وتجربته، للعلم توجد نسخة من التطبيق في المجلد رقم ٠٣٣ ضمن المرفقات.

تغيير طول المصفوفة مع المحافظة على عناصرها

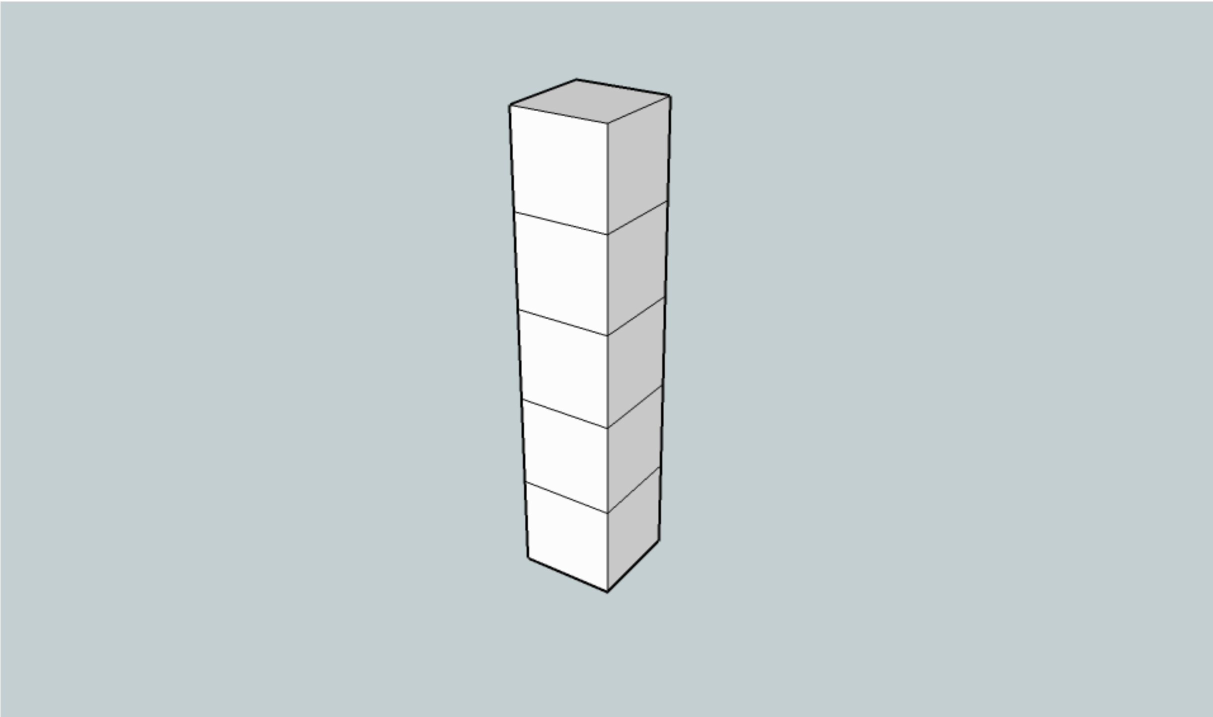
هناك ملاحظة بأنه عند استخدام ReDim ولإعادة تعريف مصفوفة تفاعلية فإن جميع عناصرها المخزنة فيها تُفقد ولا يمكن استرجاعها، وإذا أردنا تكبير أو تغيير المصفوفة مع المحافظة على عناصرها فنستخدم الكلمة Preserve بعد ReDim كالتالي: إذا كان لدينا مصفوفة باسم Authors فنقوم بتغيير طولها مع الحفاظ على عناصرها كالتالي:

```
ReDim Preserve Authors(Dim1Elements, Dim2Elements, ...)
```

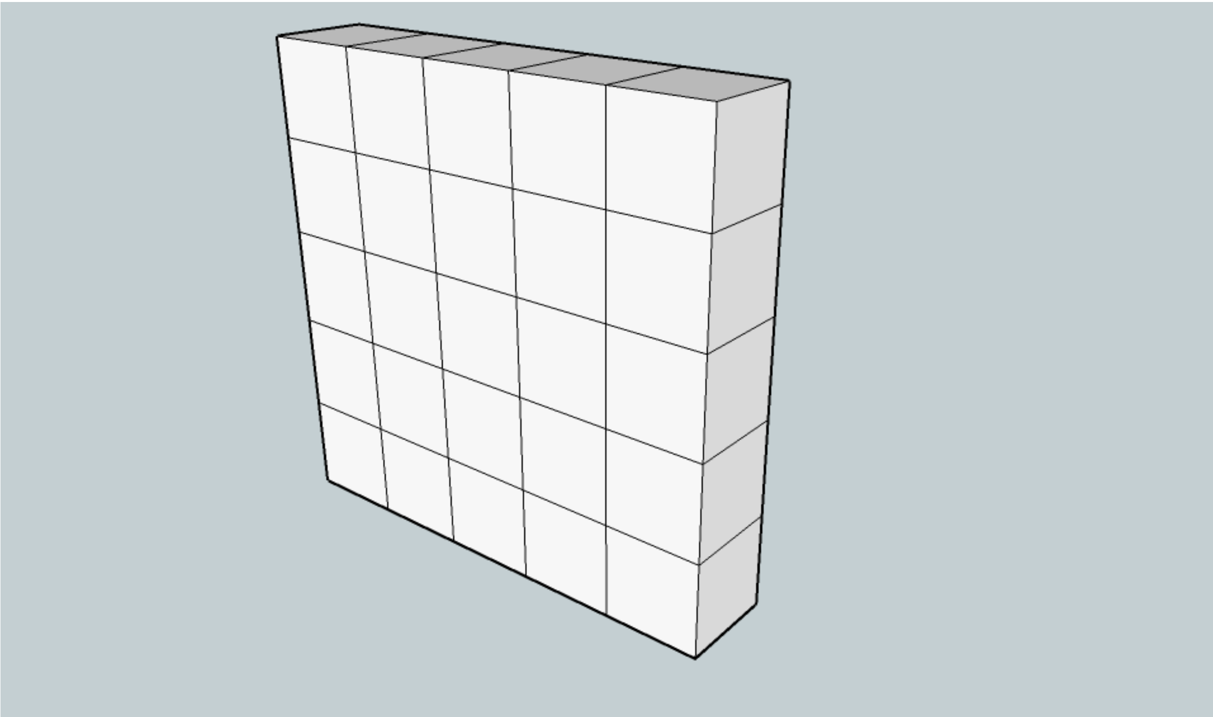
هناك ملاحظة بأننا نستطيع فقط تغيير آخر بعد في المصفوفة، فمثلاً المصفوفة ذات البعد الواحد نستطيع تغيير عدد عناصرها والمصفوفة ذات بعدين نستطيع تغيير عدد عناصر البعد الثاني فقط، أما المصفوفة ذات الثلاثة أبعاد فإننا نستطيع تغيير عدد العناصر للبعد الثالث فقط.

دعونا الآن لنتخيل المصفوفة ذات البعد الواحد وذات البعدين وذات الثلاثة أبعاد بالرسم:

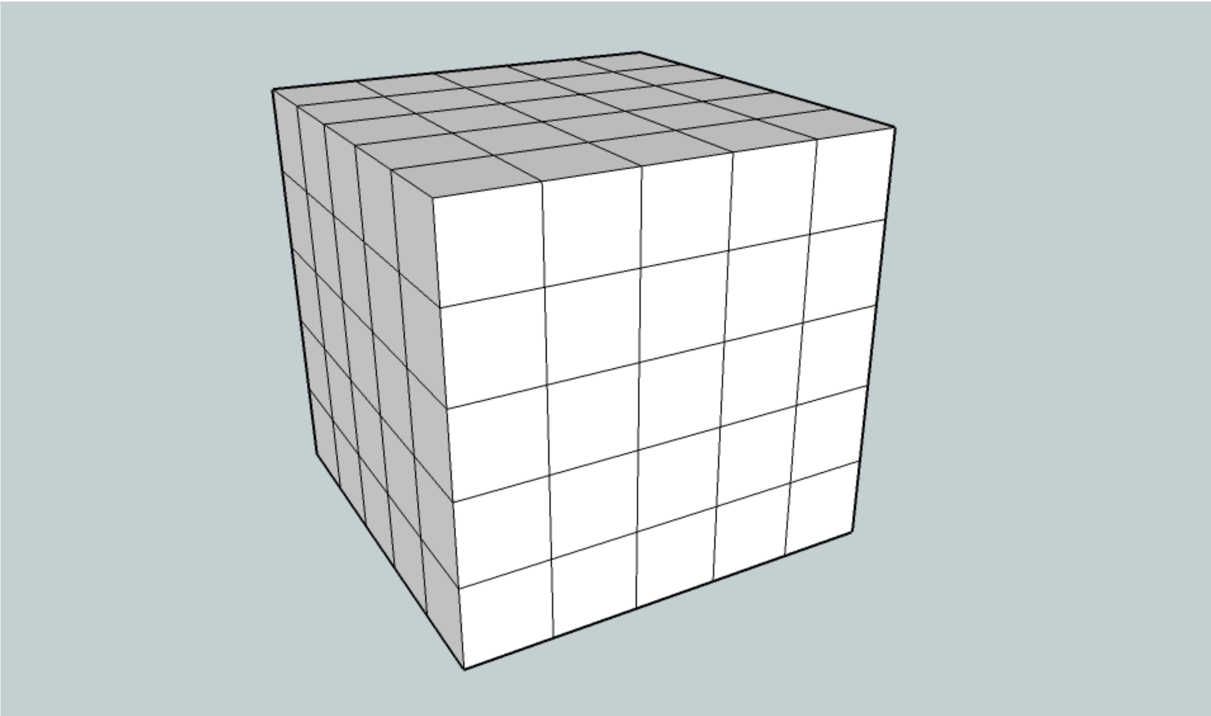
المصفوفة ذات البعد الواحد وبخمسة عناصر:



وهذا الشكل لمصفوفة ذات بعدين وبخمسة عناصر للبعد الأول وخمسة للبعد الثاني:



أما هذا الشكل فيعبر عن مصفوفة ذات ثلاثة أبعاد وفي كل بعد خمسة عناصر:



الأشكال أعلاه ستساعدك على فهم عملية تمثيل المصفوفات في ذاكرة الكمبيوتر وكيف يتعامل البرنامج معها.

خطوة إضافية: التعامل مع المصفوفات الكبيرة باستخدام الطرق Methods في كلاس المصفوفات المتوفرة ضمن بيئة التطوير:

تعلمنا في هذا الفصل كيف نصمم المصفوفات وكيف تتعامل هذه المصفوفة مع البيانات وكيف تقوم بتخزينها، في هذه الخطوة سنتعلم أكثر عن الطرق المتوفرة ضمن بيئة التطوير والتي تساعدنا على التعامل الأمثل مع البيانات الموجودة بداخل المصفوفة، حيث توفر لنا بيئة التطوير العديد من المميزات التي يمكن أن نستخدمها في التعامل مع البيانات الموجودة بداخل المصفوفة فيمكننا البحث بداخل البيانات ترتيب البيانات عكس الترتيب وتنفيذ مهام أُخرى.

في بيئة التطوير إختر Open Project ثم قم بفتح المشروع الموجود ضمن المرفق في مجلد رقم ٠٣٤ بعد فتح البرنامج ستظهر لك فورم مثل هذه:



لاحظ أسفل الفورم، حيث شريط التطور ويعرض لك هذا الشريط نسبة ما قد تم إنجازة من العملية ونسبة المتبقي. بواسطة إظهار مستطيلات خضراء في المسار الأبيض وعندما يمتلئ الفراغ الأبيض بالمستطيلات فهذا يعني إكتمال العملية. يسمى هذا الشريط Progress bar ونلاحظ هذا الشريط كثيرا عند تنزيل البرامج، الآن نستطيع إضافة مثل هذا الشريط إلى برامجنا. هناك خاصيتين هامتين لهذا الشريط وهما Maximum وتعني القيمة العليا و Minimum القيمة الدنيا ويتم ملئ الشريط بالإعتماد على هاتين القيمتين ويمكننا التحكم بهما في مرحلة الكود.

نذهب الآن إلى مرحلة الكود الموجود ضمن التطبيق المرفق وفي أول منطقة الكود نلاحظ بأننا قمنا بتعريف مصفوفة ثابتة RandArray ذات ٥٠٠ عنصر بالكود:

```
Dim RandArray(0 To 499) As Long
```

وفي حدث Load التابع لــ Form1 قمنا بتحديد قيمة الخاصيتين Maximum و Minimum التابعة للشريط ProgressBar1 بالكود التالي:

```
ProgressBar1.Minimum = 0
ProgressBar1.Maximum = UBound(RandArray)
```

أما في الحدث Click التابع للزر "تعبئة" Button1 فإن الكود الموجود فيه يقوم بتوليد أرقام عشوائية تبدأ من الواحد وتنتهي عند المائة ألف وتعبئتها في المصفوفة ثم عرض عناصر

المصفوفة في مربع النص وقد مرت علينا طريقة توليد الأرقام العشوائية في بداية هذا الكتاب (هل تتذكر كتابة برنامجك الأول) عند تطبيق الرقم المحظوظ الموجود في المرفق رقم ٠٠٢ فلا داعي لذكر الطريقة مرة ثانية.

أما بالنسبة للكود الخاص بترتيب عناصر المصفوفة فيقوم هذا الكود أولاً بتفريغ صندوق النص من محتواه ولإن المصفوفة محفوظة في الذاكرة أصلاً نقوم بترتيب عناصرها بالكود:

```
Array.Sort(RandArray)
```

بعدها نقوم بعرض عناصر المصفوفة المرتبة في صندوق النص، ونفس الشيء لترتيب عناصر المصفوفة بطريقة عكسية (راجع الكود). حيث استخدمنا:

```
Array.Reverse(RandArray)
```

بعد بتنفيذ التطبيق وقم بالضغط على الزر "تعبئة" ثم الأزرار "ترتيب" و "عكس ترتيب" واستمتع بترتيب عناصر المصفوفة وكذلك عكس ترتيب عناصر المصفوفة مع ملاحظة أن عكس ترتيب عناصر المصفوفة لا يقوم برتيب عناصر المصفوفة بشكل تنازلي وإنما يقوم بترتيبها عكس ما كانت مرتبة من قبل الضغط على الزر فإذا كانت عناصر المصفوفة مرتبة تنازلياً بعد إستخدام الأمر Array.Sort(RandArray) فيقوم بترتيبها تصاعدياً والعكس، أما إذا كانت المصفوفة غير مرتبة أصلاً فيقوم الزر "عكس ترتيب" بترتيبها بشكل عكسي بأن يجعل آخر عنصر فيها أول عنصر وقبل الأخير ثاني عنصر وعلى هذا المنوال.

عند تنفيذ البرنامج والضغط على أزرار التطبيق سنلاحظ بعض البطء في تنفيذ الأوامر وذلك بسبب أننا نقوم بتعبئة صندوق النص كل مرة يقوم فيها البرنامج بإضافة عنصر إلى المصفوفة وعند كل مرة يقوم البرنامج بترتيب عنصر أو بعكس الترتيب.

إذا أردنا أن نغير عدد عناصر المصفوفة وجعلها مصفوفة بـــ ٣٠٠٠ عنصر لملاحظة دقيقة لشريط التطور ProgressBar1 نذهب إلى أعلى الكود ونكتب :

```
Dim RandArray(0 To 2999) As Long
```

بدلاً من الكود الموجود مسبقاً وبعد تنفيذ البرنامج سنلاحظ أن شريط التطور أبطأ أكثر هذه المرة بسبب كثرة عناصر المصفوفة التي يقوم بإضافة كل عنصر جديد منها إلى صندوق النص، وكما قلنا سابقاً بأن التأخير هو بسبب تعبئة صندوق النص ثم إعادة تعبئته كل مرة نضيف فيها عنصر جديد للمصفوفة وبإمكاننا تكبير المصفوفة بتعديل الكود الموجود في أعلى منطقة الكود لعدد عناصر المصفوفة ولاخوف على بقية الكود لإننا نستخدم UBound في بقية مراحل الكود، لكن إذا زدنا عناصر المصفوفة إلى رقم معين لايمكن أن يتحمل ذلك صندوق النص TextBox بسبب قدراته المحدودة ويمكننا إستخدام RichTextBox بدلاً عنه. ولمعرفة السرعة الحقيقية التي يقوم بها البرنامج بتعبئة عناصر المصفوفة وكذالك بترتيب وعكس ترتيب عناصر المصفوفة نقوم بإضافة البادئة ' التي تقوم بإلغاء الكود التي كتبناه فيما بعدها بنفس السطر، نقوم بإضافة تلك البادئة إلى الكود الذي يأمر البرنامج بكتابة أو بترتيب أو عكس ترتيب عناصر المصفوفة وإضافتها إلى صندوق النص وبالتحديد إلى الأسطر التي تبدأ بـــ TextBox1 وبعد إضافة تلك البادئة إلى الثلاثة الأسطر الموجودة في الكود ثم تنفيذ البرنامج سنلاحظ السرعة الكبيرة التي يتم فيها تعبئة عناصر المصفوفة وكذلك ترتيب وعكس الترتيب وذلك بملاحظة سرعة تحرك شريط الـــ ProgressBar. سنلاحظ أنه حتى إذا غيرنا عدد عناصر المصفوفة إلى مائة الف عنصر سنلاحظ أن التعبئة والترتيب لا تأخذ ثوان معدودة.

خلاصة الفصل الحادي عشر

من اجل أن	قم بالتالي
إنشاء مصفوفة	نقوم بتعريف المصفوفة باستخدام الكلمة Dim كالتالي: Dim Employees(9) As String
إنشاء مصفوفة عامة	نعرفها بداخل وحدة برمجية Module بكتابة الكلمة Public كالتالي: Public Employees(9) As String
إنشاء مصفوفة عامة مع تحديد رقم أول عنصر	Public Employees(0 to 9) As String طبعاً في فيجوال بيسك ٢٠٠٨ لابد أن تبدأ المصفوفة من صفر ولكن مثل هذا

وآخر عنصر	التوضيح يسهل عملية قراءة الكود لاحقاً. مع العلم بأن هذه الطريقة غير مدعومة في نسختي الفيجوال بيسك ٢٠٠٢ و ٢٠٠٣.
تعيين قيمة لعنصر بداخل المصفوفة	نحدد اسم المصفوفة وكذلك رقم العنصر بداخل المصفوفة ثم نكتب قيمة العنصر كالتالي: `Employees(5) = "Leslie"`
إضافة Tab و Enter الى صناديق النص	نستخدم vbCrLf و vbTab لإضافة الـــ Enter للإنتقال إلى السطر التالي وللإزاحة Tab.
تعريف مصفوفة تفاعلية	المصفوفة التفاعلية هي المصفوفة التي نستطيع تغيير عدد عناصر وقت تنفيذ البرنامج ويمكننا تعريف هذه المصفوفة بتعريف مصفوفة بلشكل المعروف مع تحديد نوع عناصرها لكن مع عدم كتابة عدد العناصر، وإذا كانت المصفوفة ببعدين أو بثلاثة أبعاد فنكتب الفاصلة بداخل القوس بين كل بعد والآخر بدون كتابة عدد العناصر. ثم نقوم بإعادة تعريف المصفوفة في مرحلة الكود وإسناد عدد العناصر إلى متغير يقوم المستخدم بتعبئته في مرحلة تشغيل البرنامج كالتالي: لنفرض بأن المتغير هو Cars `ReDim Temperatures(Cars)`
التعامل مع عناصر المصفوفة	نستخدم الحلقات التكرارية للتعامل مع عناصر المصفوفة كالتالي: `Dim i As Short` `Dim Total As Single` `For i = 0 To UBound(Temperatures)` `Total = Total + Temperatures(i)`

	Next
إعادة تعريف مصفوفة مع الحفاظ على العناصر الموجودة فيها	نستخدم الكلمة Preserve بعد كلمة إعادة التعريف ReDim كالتالي: ReDim Preserve myCube(25, 25, 50)
تغيير ترتيب عناصر المصفوفة	نستخدم الطرق Methods الموجودة ضمن بيئة التطوير فمثلاً لترتيب عناصر مصفوفة اسمها RandArray ترتيباً تصاعديا نكتب الكود التالي: Array.Sort(RandArray) ولعكس الترتيب نكتب الكود: Array.Reverse(RandArray)
لإعطاء المستخدم فكرة مرئية عن عملية برمجية خاصة في الحالات التي تأخذ وقت أطول	نستخدم شريط التطور ProgressBar في برنامجنا ليأخذ المستخدم فكرة عن الوقت اللازم لتفيذ العملية سنجد الــ ProgressBar ضمن الأدوات وبالتحديد تحت قائمة Common Controls ونقوم بتحديد القيمة الدنيا والعليا له وكذلك تعبئة القيم من خلال الكود وغالباً ما نستخدم الحلقات التكرارية للتعامل معه مثل الحلقة For…Next.

الفصل الثاني عشر

التعامل مع المجوعات Collections ومجال الأسماء System.Collections

أولاً لابد أن نعرف بأن **Collections** هي عبارة عن مجموعة من الكائنات objects التي توجد في تطبيقاتنا وبشكل أوضح هي عناصر التحكم الموجودة على الفورم مثل صناديق النص والأزرار وغيرها. نحن عرفنا بأن بيئة التطوير تقوم بحفظ جميع الكائنات على الفورم مع الكود في ملف واحد ولكننا لم نعرف بأن بيئة التطوير تتعامل مع هذه الكائنات على أنها أعضاء في

مجموعه واحدة وهي Controls collection والتي تعتبر جزء من مجال الأسماء System.Collections الذي يأتي ضمن الفريم وورك التابع لمجموعة التطوير. يتم إنشاء المجموعات System.Collections أوتوماتيكياً بعد أن تقوم بإضافة فورم لبرنامجك وعندما تقوم بإضافة كائنات إلى الفورم تكون هذه الكائنات جزءاً من المجموعات System.Collections. وتقوم بيئة التطوير بالتعامل مع هذه المجموعات بنفس الطريقة التي تتعامل بها مع المصفوفات حيث أول عنصر فيها يبدأ من الصفر وهكذا. لماذا نريد أن نتعلم أكثر عن هذه المجموعات، لأننا نريد أن نتعلم كيف تتعامل بيئة التطوير مع الكائنات بداخل الفورم وكذلك نعرف أكثر إذا أردنا أن نضيف كائنات معينة (أزرار أو صناديق نص) إلى الفورم بواسطة الكود وبدون استخدام الطريقة البدائية. وتستطيع إستعراض الكائنات التي استخدمتها في برنامجك بواسطة هذه المجموعات وبدعم من بيئة التطوير.

فهرسة الكائنات في المجوعات

نستطيع فهرسة الكائنات في المجموعات Collection أو فهرسة عناصر هذه المجموعات بمعرفة رقم كل كائن في فهرس المجموعة وللعلم يقوم الفيجوال بيسك بعمل فهرس عكسي لكل كائن على الفورم فآخر كائن تمت إضافته إلى الفورم يعتبر العنصر رقم ٠ في مصفوفة مجموعة الكائنات. وعليه بمعرفة تسلسل إضافى الكائنات للفورم نستطيع فهرسة تلك الكائنات. ويمكننا بناء على ذلك كتابة أكواد لتغيير خواص هذه الكائنات وغيرها من الأكواد كالتالي:

```
Controls(0).Text = "Business"
```

فالكود أعلاه يقوم بتغيير الخاصية Text التابعة للكائن رقم ٠ (آخر كائن تمت إضافته إلى الفورم) إلى "Business"، طيب ماذا عن الكائن الذي تمت إضافته قبل الكائن الأخير يجب أن نعرف بأن هذا الكائن يأخذ التسلسل ١ في فهرس مجموعة الكائنات أما الكائن الذي تم إضافته للفورم قبل قبل الأخير فيأخذ التسلسل ٢ وهكذا لجميع الكائنات على الفورم. ومن أجل هذا المنطق يجب أن نعرف بأنه في كل مرة نضيف كائنات جديدة للفورم فإن آخر كائن يأخذ التسلسل ٠ وتتغير بقية التسلسلات برقم واحد في كل مرة نقوم بإضافة كائن جديد للفورم. نستطيع إظهار أسماء عناصر المجموعة بواسطة كود الحلقة التكرارية التالي:

```
Dim i As Integer
```

```
For i = 0 To 3
MsgBox(Controls(i).Name)
Next i
```

استخدام الحلقات التكرارية For Each...Next للتعامل مع المجموعات

نستطيع التعامل مع الكائنات بداخل المجموعات كل على حده ولكن من الأفضل التعامل معها جميعاً باستخدام الحلقات التكرارية لإننا قد نحتاج إلى تغيير أسماء الكائنات أو إلى تحريك الكائنات على الفورم أو إلى ترتيب أو تغير الأبعاد بشكل دفعه واحدة. لتنفيذ مثل هذه الأوامر نستخدم حلقة تكرارية خاصة وهي For Each...Next للتعامل مع كل الكائنات بداخل المجموعة مرة واحدة. الحلقة التكرارية For Each...Next مثل الحلقة المعروفة For...Next.

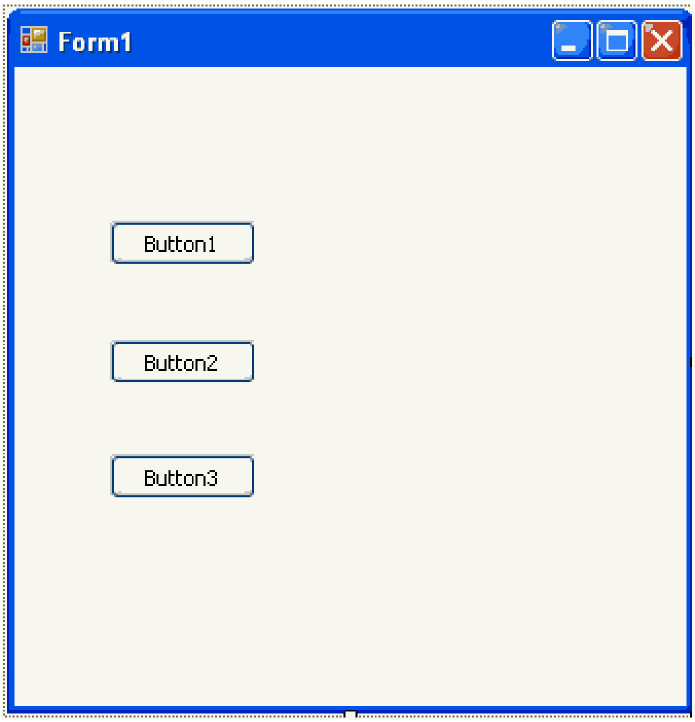
باستخدام الحلقة التكرارية For Each...Next نستطيع تعديل خواص الكائنات الموجودة ضمن المجموعة مثل إظهار أو إخفاء الكائنات وكذلك تفعيل أو إلغاء أو تحريك الكائنات أو إظهار قائمة بأسماء الكائنات وغيرها.

مثال في التعامل مع الكائنات بدخل المجموعات:

في هذا المثال سيكون معنا ثلاث كائنات علة الفورم (ثلاثة أزرار) سنتعلم كيفية التعامل مع كل هذه الأزرار سواءاً بتغيير النصوص المكتوبه عليها (تغيير الخاصية Text) وتحريك هذه الكائنات على الفورم وكذلك التعامل مع أحد هذه الكائنات بشكل مستقل عن الباقين.

١-قم بإنشاء تطبيق جديد بواسطة الفيجوال بيسك وقم بتسميته My Controls Collection.

٢–قم بإضافة ثلاثة أزرار إلى الفورم زسيكون الفورم بهذا الشكل:



٣– قم بتعديل الخاصية Name للزر الثالث Button3 إلى btnMoveObjects.

٤–قم بكتابة الكود التالي في حدث Click التابع للزر Button1 :

```
For Each ctrl In Controls
ctrl.Text = "إنقر هنا"
Next
```

الكود أعلاه يستخدم المتغير ctrl في المجموعة Controls لتغيير الخاصية Text لكل كائن على الفورم إلى "إنقر هنا" ولتعريف المتغير ctrl نذهب إلى أعلى منطقة الكود تحت Public Class Form1 ونكتب الكود :

```
Dim ctrl As Control
```

التعريف أعلاه سيسمح لنا باستخدام المتغير ctrl على طول الفورم. الآن نقوم بتنفيذ البرنامج وننقر فوق الزر الأول (ماذا نلاحظ)، مع العلم بأن تغيير الخاصية Text يمون وقت تشغيل

البرنامج فقط ولا يدوم هذا التغيير إلى بعد إغلاق البرنامج وإذا أردنا أن يستمر هذا التغيير فنستخدم قواعد البيانات لهذا الأمر وليس مجالنا الآن.

تحريك الكائنات على الفورم

لتحريك الكائنات على الفورم نضيف هذا الكود للحدث Click التابع للزر Button2 :

```
For Each ctrl In Controls
ctrl.Left = ctrl.Left + 45
Next
```

الكود أعلاه يقوم بتحريك الكائنات على الفورم بمقدار ٤٥ نقطة Pixel إلى اليمين، إذا أردنا تحريك الكائنات إلى اليسار بمقدار ٤٥ نقطة فإننا نطرح ٤٥ (نستخدم إشارة الطرح –).

في فيجوال بيسك ٦ نستخدم TWIPs بدلاً من النقطة Pixel لتحديد القياسات مع العلم بأن الــ TWIPs تساوي عُشر النقطة.

قم الآن بتشغيل البرنامج وإضغط على الزر الثاني و كرر الضغط (ماذا تلاحظ).

طبعاً قد لا تريد تحريك كل العناصر على الفورم مرة واحده وقد تحتاج تحريك كائن واحد فقط من الكائنات على الفورم، تستطيع ذلك بمعرفة اسم الكائن.

إستخدام الخاصية Name في الحلقة التكرارية For Each...Next

إذا اردنا التعامل مع عنصر أو أكثر من الكائنات الموجودة على الفورم بدون التأثير على بقية الكائنات فيمكننا الإستدلال على ذلك الكائن بالاسم حيث من المعروف لدينا بأن كل كائن على الفورم له اسم مختلف عن بقية الكائنات فيمكن استخدام هذا الاسم في مرحلة الكود ليتم التعامل معه فقط، اما إذا كان لدينا فرضاً عشرات الكائنات على الفورم، لنفرض بانه لدينا ٥٠ كائن على الفورم وأردنا تغيير خواص ١٠ فقط من هذه الكائنات فيمكننا الإستفادة من الخاصية Tag وإضافة كلمات دلالية لكل الكائنات المتشابهه في الخواص أو التي تندرج تحت مجموعة واحدة عند إرادة تغيير الخواص المراد تغييرها، فيمكننا وضع Tag واحد لهذه الكائنات ثم التعامل معها دفعة واحدة

في مرحلة الكود. طبعاً للتعامل مع الكائنات بالإستدلال بالاسم أو الكلمة الدلالية Tag لابد من استخدام أداة الشرط IF وإذا كان لدينا أكثر من شرطين يمكننا استخدام Select Case ، لنأخذ الآن تطبيق على استخدام اداة الشرط If وبدلالة الأسم Name للتعامل مع الكائنات على الفورم على نفس المثال السابق (الموجود في المرفق ٠٣٥):

نضغط Double-Click على الزر الثالث ثم نكتب الكود التالي في الحدث Click التابع للزر btnMoveObjects :

```
For Each ctrl In Controls
   If ctrl.Name <> "btnMoveObjects" Then
ctrl.Left = ctrl.Left + ٤٥
   End If
Next
```

لاحظ بأننا استخدمنا اداة الشرط لتحديد أسم الكائن على الفورم فإذا لم يكن اسمه btnMoveObjects فسيقوم البرنامج بنقله بمقدار ٤٥ نقطة.

إنشاء المجموعات الخاصة بك Your Own Collections

كما أن الفيجوال بيسك ٢٠٠٨ يقوم بإنشاء المجموعات Collections تلقائياً عند فتح الفورم وإضافة الكائنات له، تستطيع أنت أن تنشئ المجموعات الخاصة بك (ومافيش حد أحسن من حد) تستطيع إنشاء المجموعات التي تقوم بمتابعة البيانات في البرنامج والتعامل معها بشكل أوتوماتيكي وبالرغم من ان المجموعات تقوم بحفظ الكائنات تستطيع جعل مجموعاتك تحفظ الكائنات وكذلك القيم النصية والرقمية عند تنفيذ البرنامج. طبيعة هذه المجموعات هي نفس طبيعة المصفوفات التي تعرفنا عليها في الفصل السابق.

تعريف مجموعة جديدة:

يتم تعريف المجموعات الجديدة كأنها متغيرات جديدة في البرنامج، ويتم تحديد قدرات المجموعة التي قمت بتعريفها بمعرفة المكان الذي قمت بتعريف المجموعة فيه، فبإمكننا أن نقوم بتعريف هذه

المجموعات في بداية الكود للفورم بعد Public Class From1 أو في أي مكان آخر، ومن الأفضل في بداية الفورم. مثال:

```
Dim CollectionName As New Collection()
```

CollectionName هو اسم المجموعة. أما إذا قمنا بتعريف المجموعة بداخل وحدة برمجية Module فإننا نستخدم كلمة Public بدلاً عن Dim. بعد تعريف المجموعة نستخدم الطريقة Add لإضافة عناصر جديدة إلى المجموعة. ويمكننا التعاملم ع عناصر المجموعة بواسطة الحلقة التكرارية For Each...Next .

سنأخذ مثال عن طريقة التعامل مع المجموعات التي نقوم بإنشائها، في هذا المثال سنصمم تطبيق صغير يقوم بفتح وصلات لمواقع في الإنترنت وسنستخدم المجموعة التي سنُعرفها لحفظ عناوين تلك المواقع التي زُرناها، مع ملاحظة أن التطبيق لن يقوم بتصفح الإنترنت وإنما سيقوم بتحميل عنوان صفحات الإنترنت للمتصفح الموجود في جهازنا.

ننشئ تطبيق جديد بواسطة الفيجوال بيسك ٢٠٠٨ باسم My URL Collection، بعد فتح الفورم نقوم بإضافة صندوق نص وإثنين أزرار وتعديل الخواص لتكون الفورم كما في هذا الشكل:



ثم نقوم بتعريف مجموعتنا الجديدة في أعلى منطقة الكود في الفورم تحت Public Class Form1 كالتالي:

```
Dim URLsVisited As New Collection()
```

نضيف هذا الكود لزر "زيارة الصفحة":

```
URLsVisited.Add(TextBox1.Text)
System.Diagnostics.Process.Start(TextBox1.Text)
```

الكود أعلاه يقوم بإضافة الوصلة المكتوبة في صندوق النص إلى مجموعة الـــ URLsVisited وفي نفس الوقت يقوم بفتح الوصلة، وبعد كتابة العديد من الوصلات يقوم البرنامج بتسجيل الوصلات التي زرناها ويقوم بفتحها في المتصفح الموجود في جهازنا في نفس الوقت. لكن كيف نستعرض الوصلات المخزونة في المجموعة URLsVisited، نستعرض الحدث Click التابع للزر "قائمة بالمواقع" ثم نكتب هذا الكود:

```
Dim URLName As String = "", AllURLs As String = ""
For Each URLName In URLsVisited
   AllURLs = AllURLs & URLName & vbCrLf
Next URLName
MsgBox(AllURLs, MsgBoxStyle.Information, "الوصلات التي قمنا بفتحها")
```

الكود أعلاه يقوم بإسناد الوصلات إلى متغير URLName والذي بدورة ينقل هذه الوصلات جميعها إلى المتغير AllURLs بعد جعل كل وصلة في سطر، ثم نقوم باستعراضها في صندوق نص كما في السطر الأخير من الكود. MsgBoxStyle تعني نوعية صندوق الحوار الذي سيظهر وقد اخترنا النوع Information لئلا يظهر صندوق الحوار وكأنه إنذار، أما النص الذي بعد الفاصلة فهو عنوان لصندوق الحوار، الآن قم بتجربة التطبيق، نقوم بزيارة موقعين أوثلاثة ثم نستعرض هذه المواقع عن طريق الزر "قائمة بالمواقع".

نسخة من التطبيق بالمرفقات، مرفق رقم ٠٣٦.

خطوة إضافية للأمام: مجموعات الـــ VBA

لابد لنا أن نعرف معنى كلمة VBA (Visual Basic for Applications) وتعني فيجوال بيسك للبرامج وقد صممت هذه أصلاً للتعامل مع برامج الأوفس (الورد والأكسل وغيرها)، حيث يمكن تصميم برامج مصغرة (Macros) للتعامل مع الورد أو الأكسل أو لتفيذ العمليات بسهوله بواسطة تلك البرامج. فمثلاً الكود التالي لبرنامج الورد يقوم بالبحث في الملفات المفتوحه بواطة برنامج الورد فإذا وجد هذا البرنامج ملف تحت اسم myletter.doc فإنه يقوم بحفظه وإذا لم يجده يقوم بفتحه من المسار "c:\vb08sbs\chap12\myletter.doc :

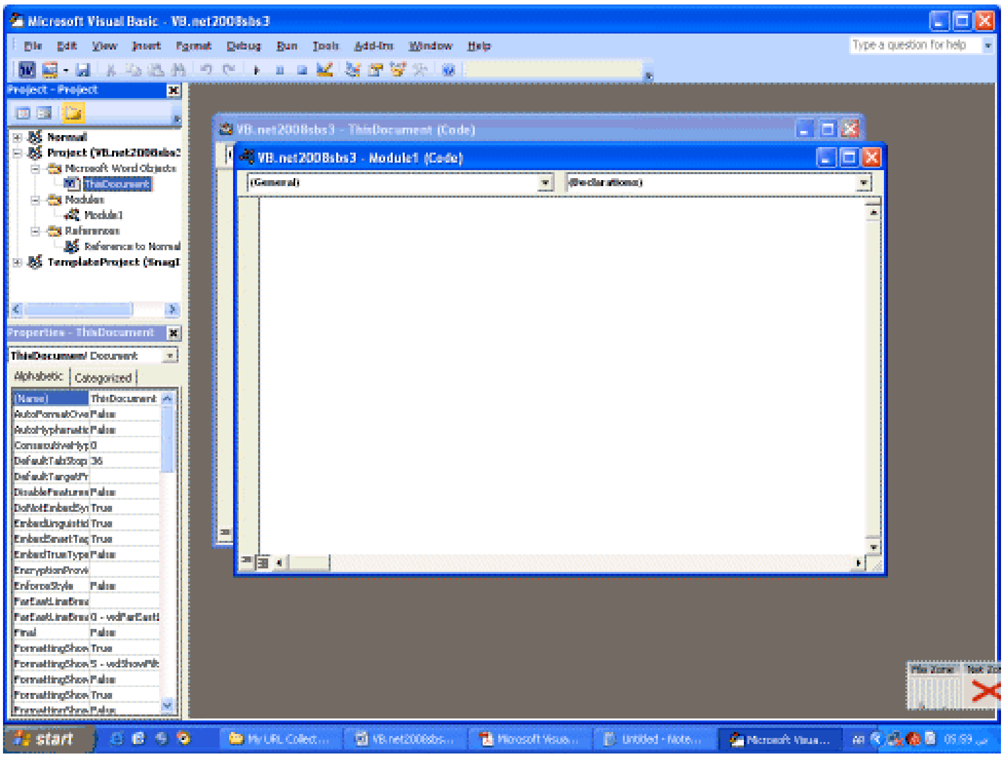
```
Dim aDoc As Document
Dim docFound As Boolean
Dim docLocation As String
docFound = False
docLocation = "c:\vb08sbs\chap12\myletter.doc"
For Each aDoc In Documents
If InStr(1, aDoc.Name, "myletter.doc", 1) Then
docFound = True
aDoc.Save
Exit For
End If
Next aDoc
If docFound = False Then
Documents.Open FileName:=docLocation
End If
```

الكود أعلاه يقوم بتعريف المتغيرات aDoc، docFound، docLocation ثم قمنا بالبحث في الملفات المفتوحة عن الملف myletter.doc بواسطة الأمر InStr والذي يبحث عن متغير نصي بداخل متغير نصي آخر وإذا وجده فإنه سيقوم بحفظه بالكود aDoc.Save ثم قمنا بإغلاق

الحلقة التكرارية For Each بالكود Exit For حتى لاتذهب بنا بعيداً وتنتج لنا نتائج غير مطلوبة، أما إذا لم نجد الملف myletter.doc فإن البرنامج سيقوم بفتحة من المسار المحدد له في الكود بالأعلى. وللعلم هنا يقوم برنامج الورد وبرامج الأوفس بشكل عام بالتعامل مع ملفاته بشكل المجموعات Collections الموجودة في برنامج الفيجوال بيسك ٢٠٠٨ (فإذا فتحنا فورم جديد يقوم الفيجوال بفتح مجموعة جديدة وإذا أضفنا لها كائنات جديدة يقوم الفيجوال بيسك بإضافة هذه الكائنات إلى المجموعة) فكل الملفات المفتوجة موجودة في مجموعة Collection خاص ببرنامج الورد وكل النصوص والفقرات كذلك. وهناك العديد من البرامج الأُخرى التي تدعم الـــVBA تقوم بنفس العمل.

إدخال الكود إلى برنامج الورد

الكود بالأعلى لا يمكن فتحة عن طريق برنامج الفيجوال بيسك وإنما لابد من فتحة ببرنامج الورد، وتختلف طريقة فتحه ببرنامج الورد بحسب نسخة الورد المتوفرة لدينا، فإذا كانت لديك نسخة الورد ٢٠٠٣ فإن طريقة إضافة الكود إلى الورد تكون بأن نذهب إلى القوائم **Tools** (الأدوات) ثم منها إلى (الماكرو) Macros ثم Create New Macro أو إضافة ماكرو جديد قم بكتابة اسم للماكرو سيفتح لك البرنامج محرر الفيجوال بيسك التابع للورد قم بإضافة الكود في المحرر. أما إذا كنت تستخدم نسخة ٢٠٠٧ من الأوفس فاختر قائمة المطورين **Developer** ثم منها اختر Macro (ماكرو) ثم اختر اسماً جديداً للماكرو وقم بكتابة كود الماكرو في محرر الفيجوال بيسك الذي سيظهر لك، إذا لم تظهر لك قائمة المطورين **Developer** فقم بإظهارها من خيارات الورد Word Options ، انظر لشاشة الفيجوال بيسك المرفقة مع برنامج الورد **Microsoft Word**



بعد كتابة الكود تستطيع تنفيذة بنفس الشكل الذي نقوم فيه بتنفيذ الكود في فيجوال بيسك، مع ملاحظة أن أكواد الماكرو تعمل على كل إصدارات الأوفس لكن في بعض الحالات النادرة قد تصادفك بعض الأخطاء في حالة تطبيق الكود على أكثر من نسخة من نسخ الأوفس، وقد تحتاج لتعديل الكود قليلاً ليتناسب مع نسخة الأوفس.

خلاصة الفصل الثاني عشر

من اجل أن	قم بالتالي
التعامل مع عناصر المجموعات Collection	نستخدم الحلقات التكرارية التي تقوم بالتعامل مع كل عنصر في المجموعات Collection على حده. مثال: Dim ctrl As Control For Each ctrl In Controls ctrl.Text = "Click Me!"

	Next
تحريك بعض الكائنات على الفورم من اليسار إلى اليمين	باستخدام الحلقات التكرارية للتعامل مع الكائنات الموجودة بداخل المجموعة نقوم بتعديل الخاصية Left التابعة للكائنات كالتالي: Dim ctrl As Control For Each ctrl In Controls ctrl.Left = ctrl.Left + 25 Next
التعامل مع كائن معين ضمن مجموعة الكائنات	باستخدام الخاصية Name والحلقات التكرارية وأداة الشرط If نقوم بالتعامل مع الكائن المراد التعامل معه كالتالي: Dim ctrl As Control For Each ctrl In Controls If ctrl.Name <> "btnMoveObjects" Then ctrl.Left = ctrl.Left + 25 End If Next
تعريف مجموعة جديدة وإضافة عناصر جديدة لها	نحتاج لتعريف مجموعات جديدة بسبب سهولة التعامل مع المجموعات حيث أنها بخصائص المصفوفات ونستطيع حفظ بيانات نصية وغيرها بداخلها، لذلك نعرف مجموعة جديدة كالتالي: Dim *CollectionName* As New Collection() حيث *CollectionName* هي اسم المجموعة ويمكن إضافة عناصر جديدة

	للمجموعة بالطريقة Add كالتالي: *CollectionName*.Add(TextBox1.Text)
إستخدام الـــ VBA فيجوال بيسك للبرامج	إذا كانت لديك نسخة الورد ٢٠٠٣ فإن طريقة إضافة الكود إلى الورد تكون بأن نذهب إلى القوائم Tools (الأدوات) ثم منها إلى (الماكرو) Macros ثم Create New Macro أو إضافة ماكرو جديد قم بكتابة اسم للماكرو سيفتح لك البرنامج محرر الفيجوال بيسك التابع للورد قم بإضافة الكود في المحرر. أما إذا كنت تستخدم نسخة ٢٠٠٧ من الأوفس فاختر قائمة المطورين Developer ثم منها اختر Macro (ماكرو) ثم اختر اسماً جديداً للماكرو وقم بكتابة كود الماكرو في محرر الفيجوال بيسك الذي سيظهر لك، إذا لم تظهر لك قائمة المطورين Developer فقم بإظهارها من خيارات الورد Word Options برنامج الورد يتعامل مع الكثير من المجموعات Collections من ضمنها ملفات الورد والفقرات ضمن ملفات الورد.

الفصل الثالث عشر: إستعراض الملفات النصية والتعامل معها

للتعامل مع الملفات النصية يجب أن نعرف أربع دوال تسهل لنا عملية التعامل مع تلك الملفات، هذه الدوال هي:

FileOpen وتقوم هذه الدالة بفتح الملف النصي.

LineInput تقرأ سطر من سطور الملف النصي.

EOF تبحث عن نهاية الملف النصي.

FileClose تقوم بإغلاق الملف النصي.

من أسهل الطرق لفتح الملفات النصية هي فتح الملف النصي بداخل صندوق النص، وإذا كان الملف النصي كبيراً نضيف أشرطة تمرير ScrollBars لصندوق النص حتى يستطيع المستخدم قراءة الملف النصي كاملاً.

فتح الملفات النصية:

للملفات النصية بصمة مميزة وتحتوي على مجموعة من النصوص والاسطر والكلمات تنتهي هذه الملفات بلاحقة txt، ini، log، inf وتحتوي الملفات على نصوصة مرتبة بشكل مميز نستطيع قراءتها من خلال صناديق النص. لفتح الملفات النصية نستخدم **OpenFileDialog** نافذة الحوار التي تفتح لنا الملفات، ثم نقوم بوضع فلتر لتحديد نوع الملف النصي، كأن نحديد الفلتر **txt** لتقوم النافذة بإظهار الملفات النصية فقط، ثم نختار الملف المحدد ونوافق عليه، فتقوم النافذة OpenFileDialog بحفظ مسار الملف لنا نستطيع استخدام هذا المسار لفتح الملف، بعبارة أُخرى نافذة الحوار OpenFileDialog لا تفتح الملف وإنما تعطينا مسار الملف.

الدالة FileOpen

بعد معرفة مسار الملف النصي بواسطة الخطوة السابقة نقوم باستخدام الدالة FileOpen لفتح الملف والطريقة العامة لهذه الدالة كالتالي:

FileOpen(filenumber, pathname, mode)

حيث filenumber هو رقم الملف ويكون من ١ إلى ٢٥٥.

و pathname هو مسار الملف الذي اخترناه في الخطوة السابقة.

أما mode فيعني (وضع الإستخدام) كيف سنتخدم هذا الملف وسنعرف عليها لاحقاً، من المهم أن نعرف بأننا سنستخدم وضعين للملفات وهما وضع للمدخلات OpenMode.Input ووضع للمخرجات OpenMode.Output.

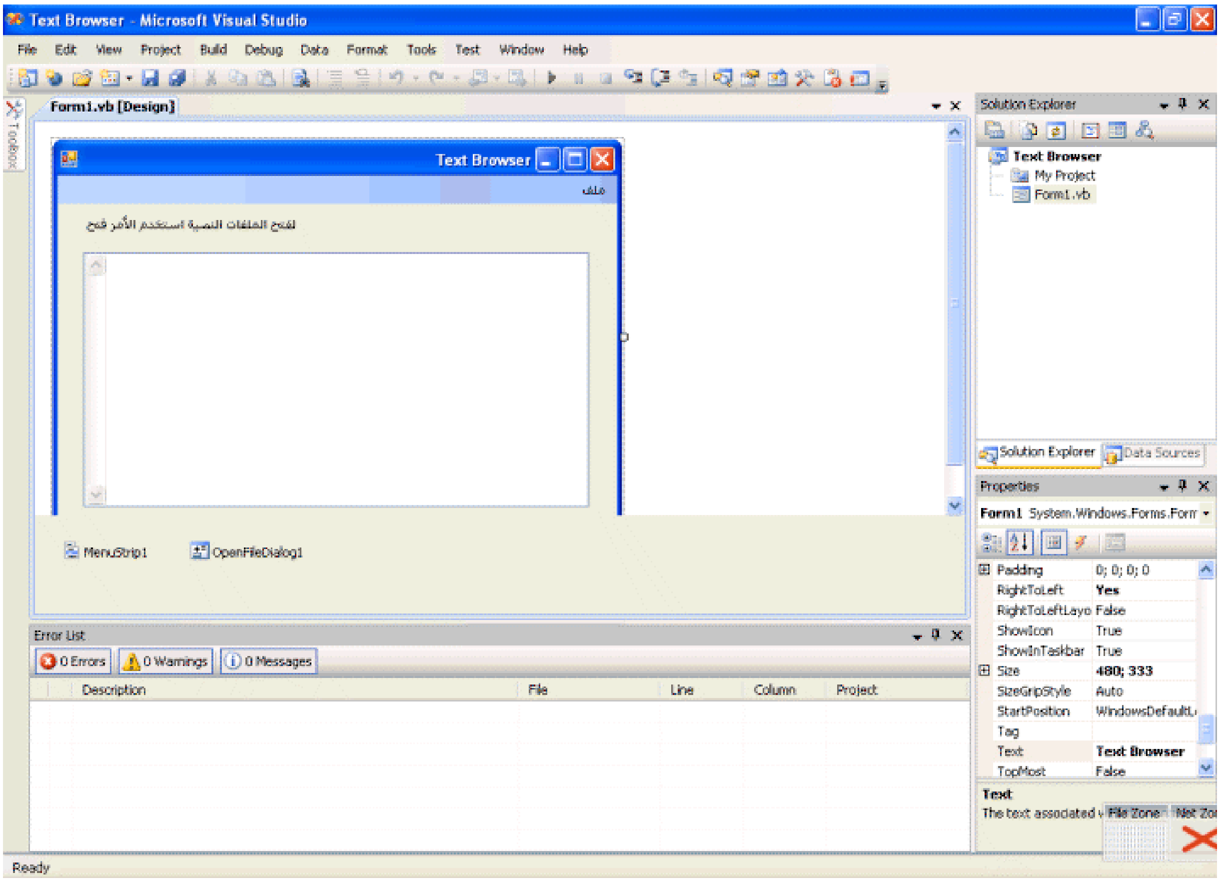
لايوجد شئ مميز عند تعيين رقم للملف، إلا في حالة واحده: إذا كان لدينا أكثر من ملف مفتوح

وأردنا من برنامجنا أن يتعامل مع ملف واحد من هذه الملفات، فيمكننا ذلك بدلالة الرقم. معظم دوال فتح الملفات تستخدم الصيغة:

```
FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
```

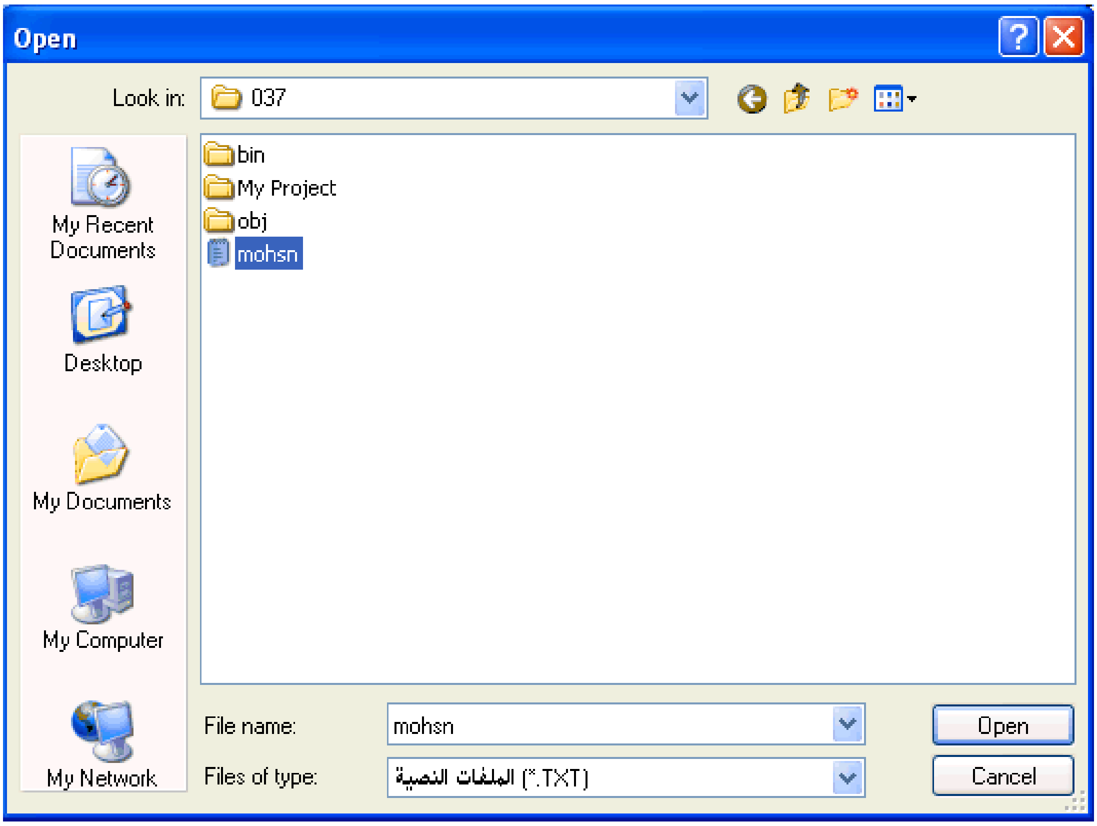
فالرقم ١ ومسار الملف هو المسار المختار في نافذة الحوار OpenFileDialog1 أما الوضع فهو وضع الإدخال OpenMode.Input، عند التعامل مع الملفات النصية يجب علينا التعامل معها بطريقة تمسى الطريقة السياقيه sequential لإن طبيعة الملفات النصية تحتم علينا التعامل معها على حسب ترتيب السياق، أما في قواعد البيانات فنستطيع التعامل معها بأي ترتيب وبأي شكل.

نقوم الآن بفتح تطبيق الملفات النصية والموجود في المرفق رقم ٠٣٧. بعد فتح التطبيق وإظهار الفوم يظهر لنا الشكل التالي:



يحتوي النموذج على صندوق نصي كبير، وكذلك على قائمة منسدلة تحتوي على فتح، علق، خروج. أما الكائنات الأخرى الموجودة على الفورم فهي نافذة فتح الملفات OpenFileDialog1

ولبيل يحتوي على تعليمات نصية. نقوم الآن بتنفيذ البرنامج بالضغط على F5 وسيفتح لنا النموذج، من النموذج (الفورم) المفتوح نختار ملف ثم فتح سيفتح لنا نافذة استعراض الملفات وسيقوم بإظهار الملفات النصية بلاحقة txt بسبب الفلتر الذي طبقناه في مرحلة الكود (سنتعرف عليه).



نقوم باختيار الملف mohsn المرفق بالتطبيق ثم نختار Open سيقوم التطبيق بفتح الملف المختار، وهو عبارة عن جزء من قصيدة للشهيد محمد محمود الزبيري، في صندوق النص بداخل النموذج (الفورم). كالتالي:



الآن سنقوم بمراجعة الكود المستخدم، نفتح النموذج (الفورم) ثم Double-Click على فتح سنجد الكود التالي تحت الحدث Click التالي لـــ OpenToolStripMenuItem:

```
Dim AllText As String = "", LineOfText As String = ""
OpenFileDialog1.Filter = "الملفات النصية (*.TXT)|*.TXT"
OpenFileDialog1.ShowDialog() 'display Open dialog box
If OpenFileDialog1.FileName <> "" Then
    Try 'open file and trap any errors using handler
        FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
        Do Until EOF(1) 'read lines from file
            LineOfText = LineInput(1)
            'add each line to the AllText variable
            AllText = AllText & LineOfText & vbCrLf
        Loop
        lblNote.Text = OpenFileDialog1.FileName  'update label
        txtNote.Text = AllText 'display file
        txtNote.Enabled = True 'allow text cursor
        CloseToolStripMenuItem.Enabled = True  'enable Close command
        OpenToolStripMenuItem.Enabled = False  'disable Open command
    Catch
        MsgBox("حدث خطأ ما.")
    Finally
        FileClose(1) 'close file
    End Try
End If
```

في السطر الأول تعريف المتغيرات، السطر الثاني تعيين الفلتر txt للنافذة اختيار الملفات OpenFileDialog1، وفي السطر الثالث أمر بفتح نافذة البحث عن الملفات، ومن السطر الرابع وما بعده استخدمنا اداة الشرط IF لتحديد ما إذا كان المستخدم قد اختار ملف معين فإذا قد اختار ملف فسوف تقوم نافذة اختيار الملفات بتحويل المسار إلى المعامل FileName وعليه إذا كان

FileName لايساوي "" أي أنه (FileName أو المسار) ليس فارغاً سيقوم البرنامج بتحويل الأمر إلى السطر الذي يليه والذي يقوم باستخدام الداله FileOpen لفتح الملف النصي المختار.

وسيقوم البرنامج بالقراءة حتى السطر الأخير من الملف، بقية الكود تقوم بتفعيل صندوق النص، وكذلك بكتابه مسار الملف في الليبل في أعلى النموذج (الفورم)، وفي نفس الوقت تقوم بتفعيل الزر (إغلاق) الموجود في القائمة المنسدلة، والذي يقوم بإغلاق الملف النصي المفتوح بدون إغلاق البرانامج كاملاً. قم بقراءة الشفرة مرة ثانية لتفهم منها الكثير.

ننزل قليلاً في منطقة الكود لنرى الكود الخاص بزر القائمة "إغلاق"، وهو الزر الذي يقوم بإغلاق الملف النصي الذي فتحناه، والموضوع تحت الحدث Click لنرى هذا الكود:

```
txtNote.Text = ""                'clear text box
lblNote.Text = ".فتح الأمر استخدم النصية الملفات لفتح"
CloseToolStripMenuItem.Enabled = False  'disable Close command
OpenToolStripMenuItem.Enabled = True    'enable Open command
```

السطر الأول يقوم بمسح كل النصوص من صندوق النص، بعد ذلك نقوم بتعبئة الليبل الموجود في النموذج (الفورم) بالعبارة "استخدم الأمر فتح لفتح الملفات النصية" بدلاً من المسار التابع للملف النصي الذي فتحناه قبل قليل. السطر الثالث من الكود يقوم بإلغاء تفيل الزر "إغلاق" في القائمة لإننا لن نحتاجه إذا لم يكن موجود معنا ملف نصي مفتوح. أما السطر الرابع فيفعل لنا الزر "فتح" في القائمة لنقوم بفتح ملف نصي آخر.

استخدام الفئة (الكلاس) StreamReader و My.Computer.FileSystem لفتح الملفات النصية:

تعلمنا في الدرس السابق كيف نقوم بفتح الملفات النصية باستخدام الدالة FileOpen التابعة للفيجوال بيسك، الآن نحن بصدد فئة ومجال أسماء يقومون بنفس العمل (فتح الملفات النصية) وهما الفئة (الكلاس) StreamReader ومجال الأسماء My.Computer.FileSystem الفرق الوحيد هنا هو أن الفئة ومجال الأسماء المذكورين يتبعان إطارات عمل الدوت نت بشكل مباشر وليسوا خاصين بالفيجوال بيسك فقط، لذلك نستطيع استخدام هاتين الطريقتين في أية لغة برمجة

تابعة للدوت نت (السي شارب، السي + +، وغيرها)، معرفة الكثير من الطرق التي تؤدي إلى نفس العمل يفيدنا كثيراً لاحقاً، وسيكون لدينا الخيار الذي نريد في حالة أردنا فتح الملفات النصية.

الفئة (الكلاس) StreamReader

هذه الفئة تقوم بقراءة الملفات النصية، وسنقوم باستخدامها كثيراً في هذا الكتاب وستجدها مثلاً في الفصل السادس عشر من هذا الكتاب، تتبع هذه الطريقة إطارات عمل الدوت نت، ولاستخدامها لا بد من إستيراد مجال الأسماء الخاص بها في أول منطقة الكود في الفورم وقبل أي سطر آخر للكود، نكتب هذا الكود:

```
Imports System.IO
```

تعلمنا طريقة كتابة أكواد استيراد مجالات الأسماء في الفصل الخامس من هذا الكتاب. بعد كتابة كود الإستيراد بالأعلى وأردنا فتح ملف نصي بداخل صندوق نص في برنامجنا نضيف هذا الكود ولنفرض بأن لدينا الكائن TextBox1 على الفورم والملف النصي هو mohsn.txt :

```
Dim StreamToDisplay As StreamReader
StreamToDisplay = New StreamReader("C:\ mohsn.txt")
TextBox1.Text = StreamToDisplay.ReadToEnd
StreamToDisplay.Close()
TextBox1.Select(0, 0)
```

الكود يفترض بأن الملف موجود في :C لكننا نستطيع تعديل الكود ونكتب المسار الصحيح للملف، وفي نفس الوقت نستطيع استخدام نافذة الحوار OpenFileDialog ونسمح للمستخدم من اختيار الملف ثم الإستفادة من المعامل FileName التابع لنافذة الحوار المذكورة.

مجال الأسماء My

الطريقة الثانية لفتح الملفات النصية هي استخدام مجال الأسماء My، تمت إضافة مجال الأسماء My إلى إطارت عمل الدوت نت في عام ٢٠٠٥ لتسهيل البرمجة ويقوم هذا المجال بالتعامل مع النماذج (الفورم) وكذلك إظهار معلومات عن ملف في الكمبيوتر أو عن مستخدم الكمبيوتر، أو عن

البرنامج نفسه، واستخدام بعض الخدمات على الإنترنت، يعزف الكثير من المبرمجين عن استخدام هذا المجال بسبب إعتقادهم صعوبة التعامل معه، وبارغم أنه أضيف ليسهل عملية البرمجة، وللتوضيح فمجال الأسماء My ينقسم إلى أكثر من مجموعة كالتالي:

My.Application	يعطينا معلومات لها علاقة ببرنامجنا، مثل مسار البرنامج ونسخة البرنامج والعنوان وغيرها.
My.Computer	يعطينا معلومات عن العتاد والبرامج ، والملفات الموجودة محلياً، ونسفيد من هذه المجموعة لفتح الملفات النصية.
My.Forms	يعطينا معلومات عن النماذج بداخل برنامجنا، سنعرف أكثر عن هذه المجموعة في الفصل السادس عشر من هذا الكتاب بعون الله.
My.Resources	يعطينا معلومات عن المراجع والدوال والمكتبات التي استند عليها برنامجنا (للإطلاع فقط).
My.Settings	يعطينا معلومات عن الإعدادات في برنامجنا ونستفيد من ذلك بخزن وإعادة الإعدادت لبرنامجنا بطريقة اوتوماتيكية.
My.User	معلومات عن المستخدم الحالي.
My.WebServices	معلومات عن خدمات الويب الحالية، واساساً يمكن الإعتماد علية لفتح خدمات ويب جديدة.

مجال الأسماء My يعتبر من أسهل الطرق البرمجية للوصول إلى الهدف، وعند كتابة الأكواد التابعة له يقوم الإكمال التلقائي IntelliSense بتسهيل المهمة أكثر.

نستطيع استخدام مجال الأسماء My.Computer.FileSystem بالإضافة إلى الطريقة ReadAllText لفتح الملفات النصية واتعراض محتوياتها بداخل صندوق نص، هنا الكود الذي نستخدم فيه مجال الأسماء المذكور لفتح الملفات النصية، على إعتبار بأن لدينا صندوق نص باسم txtNote مثل المثال السابق، نستطيع استخدام المثال السابق مع هذا الكود:

```
Dim AllText As String = ""
OpenFileDialog1.Filter = "Text files (*.TXT)|*.TXT"
OpenFileDialog1.ShowDialog() 'display Open dialog box
If OpenFileDialog1.FileName <> "" Then
AllText =
My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
txtNote.Text = AllText 'display file
End If
```

ولإننا استخدمنا الطريقة ReadAllText لفتح الملف النصي كاملاً بداخل صندوق النص، تعتبر هذه الطريقة من أسرع الطرق لفتح الملفات النصية وأسرع من الطريقة الأولى خاصة، وهنا ملاحظة بأن مجال الأسماء My يعتبر سريع جداً للتعامل مع العديد من الأمور البرمجية، لكن الطريقتين السابقتين لهما ميزات للتعامل مع المحتوى النصي أفضل من ميزات مجال الأسماء My، لإن مجال الأسماء يتعامل مع الملف ككل وكوحدة واحدة ويقوم بفتحة كاملاً، لذلك فهو أسرع، لكن طريقتي StreamReader و FileOpen تتعامل مع الملفات النصية بشكل سطري لذلك فميزاتهما النصية أفضل بكثير، خاصة إذا أردنا استخدام الترتيب التصاعدي وغيره من الميزات التي سنحتاجها لاحقاً، وعلية لابد من فهم الثلاث الطرق للتعامل مع الملفات النصية وتحديد الطريقة الأفضل في الوقت المناسب بحسب الحاجة التي نحتاجها من فتح الملف النصي.

إنشاء ملف نصي جديد على الهارد

قد نحتاج لإنشاء ملف نصي وحفظة على الهارد للعديد من الأسباب منها حفظ إعدادت برنامجنا، أو إعداد تقارير معينة، أو تصميم برنامج للتعامل مع النصوص مثل (Word)، أما طريقة حفظ النصوص في ملف نصي فتكون كالتالي:

١- أخذ المدخلات النصية من المستخدم وإسنادها إلى متغيرات معينة.

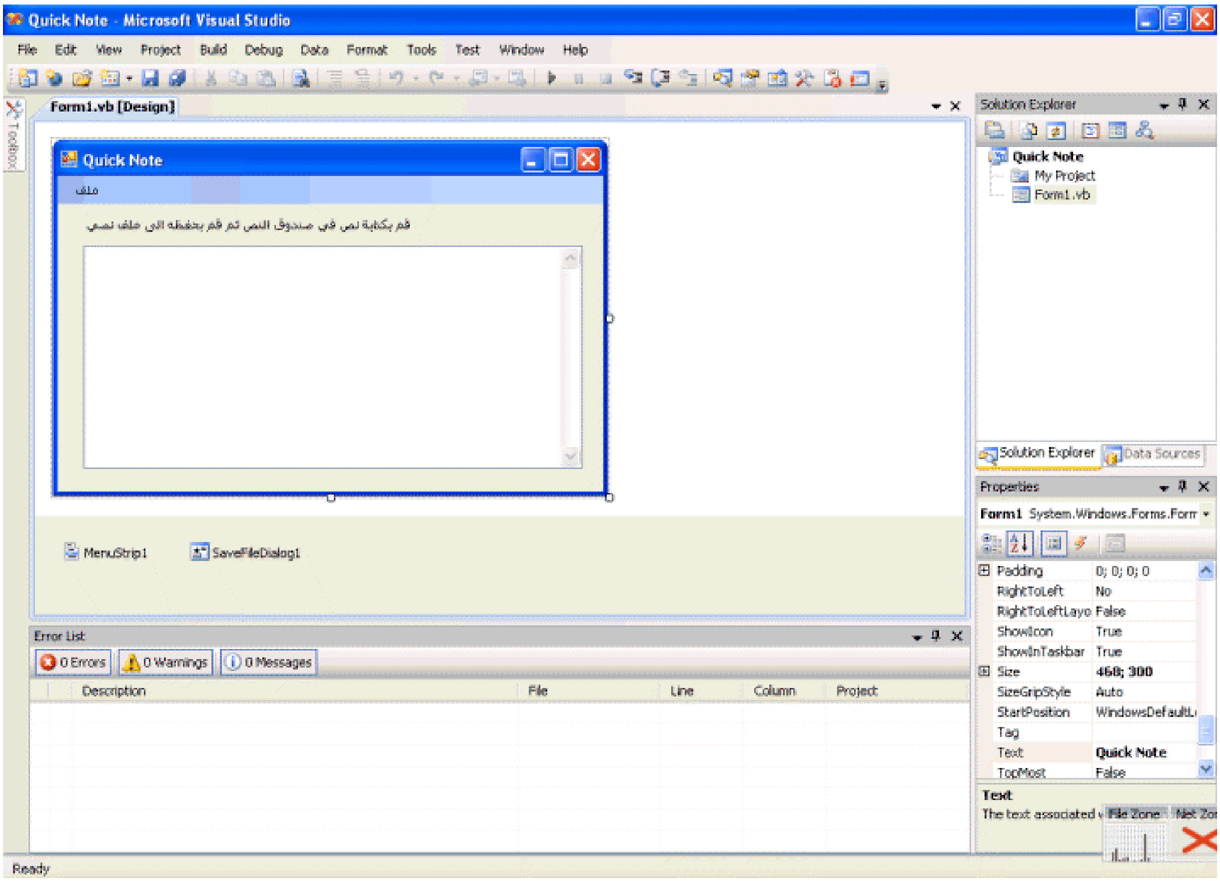
٢- السماح للمستخدم من تحديد مكان حفظ الملف النصي باستخدام نافذة حوار SaveFileDialog.

٣- استخدام المسار الذي يحدده المستخدم لحفظ البيانات النصية فيه لفتح الملف.

٤- استخدام الدالة PrintLine لحفظ البيانات النصية إلى الملف المفتوح.

٥- بعد إكمال الحفظ نقوم بإغلاق الملف المفتوح بواسطة الدالة File.Close.

قم بفتح الملف رقم ٠٣٨ من المرفقات لتجد تطبيق باسم Quick Note قم بفتحه ثم استعرض النموذج (الفورم كالتالي):



كما يلاحظ من الصورة هذا المشروع، تقريباً نفس المشروع السابق، فقط نحن استبدلنا OpenFileDialog بـ SaveFileDialog وقمنا بتعديل الكود وبنود القائمة. نقوم بتنفيذ البرنامج ونكتب نص في صندوق النص ثم نختار حفظ باسم، نلاحظ ظهور نافذة حوار لحفظ الملف النصي نختار مسار ثم نقوم بحفظ الملف النصي بلاحقة txt.

مراجعة الكود:

الكود الخاص بإدخال التاريخ إلى النص هو:

```
txtNote.Text = My.Computer.Clock.LocalTime & vbCrLf &
txtNote.Text
txtNote.Select(1, 0)  'remove selection
```

السطر الأول من الكود يستخدم مجال الأسماء My لإدخال الوقت إلى صندوق النص، ومع الأخذ في الإعتبار وجود نص سابق في صندوق النص قمنا بإضافة النص السابق إلى التاريخ الذي أضفناه، بإمكاننا إضافة التاريخ بطرق أُخرى بدون استخدم مجال الأسماء My. أما الكود الخاص بحفظ الملف النصي فهو:

```
SaveFileDialog1.Filter = "الملفات النصية (*.txt)|*.txt"
SaveFileDialog1.ShowDialog()
If SaveFileDialog1.FileName <> "" Then
   FileOpen(1, SaveFileDialog1.FileName, OpenMode.Output)
   PrintLine(1, txtNote.Text)  'copy text to disk
   FileClose(1)
End If
```

السطر الأول يقوم بوضع فلتر للملفات ويسمح فقط للنوع النصي بلاحقة txt فقط، أما السطر الثاني فيظهر لنا نافذة حوار لحفظ الملفات، وعند إختيار المستخدم لمسار معين كما في بقية الكود يقوم البرنامج بفتح الملف النصي الذي إختاره المستخدم (أو يقوم بإنشاءه إذا لم يكن موجوداً)، ثم يقوم البرنامج بحفظ النص الموجود في صندوق النص بداخل الملف النصي باستخدام الداله PrintLine الموضحه في الكود والتي تقوم بحفظ النص الموجود في صندوق النص. بعدها مباشرة نستخدم الدالة FileClose لغلق الملف النصي.

التعامل مع النصوص ضمن الملفات النصية:

تعلمنا في المثالين السابقين كيف نقوم بفتح ملف نصي وكيف نقوم بحفظ النصوص في ملف نصي، الآن سنتعرف إلى ميزات جميلة جداً في طرق التعامل مع الملفات النصية، هذه الطرق ستفيدنا كثيراً في إستخراج معلومات من الملفات النصية، كذلك في ترتيب محتويات الملفات النصية (تصاعديا، تنازلياً).

دعونا الآن نتخيل كيف يقوم برنامجنا بترتيب معين (تصاعدي، أو تنازلي) لعدد كبير من الكلمات في مصفوفة أو في عمود معين أو في ملف نصي: يقوم البرنامج بترتيب الكلمة الأولى من القائمة ثم يأخذ الكلمة الثانية ويقارناه مع الأولى لتحديد هل هي قبل أو بعد ثم يأخذ الكلمة الثالثة ويقارنها ويحدد مكانها، قبل الأولى، قبل الثانية أو بعدها ويستمر في المقارنة حتى نهاية قائمة الكلمات، ما يهمنا هنا الوقت اللازم لتنفيذ العملية، طبعاً علماء الكمبيوتر دائماً يحسبون للوقت ألف حساب وهناك العديد من الدوال والخوارزميات التي تسهل عملية الفهرسة. سوف نتعلم هنا بعض الأساسيات في مقارنة النصوص بهدف الفهرسة، ستفيدنا هذه الأساسيات في فهرسة قوائم الكلمات أو في فهرسة النصوص وغيرها.

الفئة (الكلاس) String

للتعامل الأمثل مع النصوص نحتاج للفئة String ، يمكننا القول بأننا لم نستخدم أية معامل أو دالة للتعامل مع النصوص لحد الآن عدا استخدام أداة الربط & للربط بين النصوص وإضافة النصوص بعضها إلى بعض. فإذا كان لدينا نص مفرق ونريد أن نربطه مع بعضه نستخدم أداة الربط & كالتالي:

```
Dim XYZ as String
XYZ = TextBox1.Text & " اسم الطالب"
```

يقوم الكود أعلاه بإضافة الكلمة "اسم الطالب" إلى النص المكتوب في صندوق النص TextBox1.

أما عند التعامل مع النصوص باستخدام الفئة String فالأمر يختلف قليلاً، فالفئة المذكورة تسهل لنا الكثير من عمليات التعامل مع النصوص، فعملية ربط النصوص مثلاً تكون كالتالي:

```
Dim XYZ as String
```

XYZ = String.Concat(TextBox1.Text & " اسم الطالب")

فيجوال بيسك يعتمد كلا الطريقتين للربط بين النصوص، كما أن فيجوال بيسك يعتمد الطرق القديمة والخاصة بالنسخ الأقدم من فيجوال بيسك في التعامل مع النصوص، فمثلاً الفيجوال بيسك ٢٠٠٨ يعتمد الدوال (Mid, UCase, LCase) وغيرها من الدوال وفي نفس الوقت يعتمد الدوال الجديدة للتعامل مع النصوص والتي تعتبر من ميزات إطارات العمل (الدوت نت) حيث يمكن تطبيقها تحت كل لغات البرمجة المدعومة من الدوت نت، ومن هذه الدوال (Substring, ToUpper, ToLower). الجدول التالي يوضح بعض الطرق Methods القديمة والجديدة للتعامل مع النصوص مع أمثله:

طريقة الدوت نت	دالة الفيجوال بيسك	الوصف	مثال
ToUpper	UCase	تحويل الحروف من صغيرة إلى كبتل (كبيرة)	Dim Name, NewName As String Name = "Kim" NewName = Name.ToUpper 'النتيجة ستكون = "KIM"
ToLower	LCase	تحويل الحروف من كبيرة إلى صغيرة	Dim Name, NewName As String Name = "Kim" NewName = Name.ToLower 'النتيجة ستكون = "kim"
Length	Len	لتحديد عدد الحروف في النص	Dim River As String Dim Size As Short River = "Mississippi" Size = River.Length 'النتيجة ستكون = 11
Substring	Mid	ترجع لنا نص معين ابتداء من النقطة التي نريد أن تبدأ منها مع العلم	Dim Cols, Middle As String Cols = "First Second Third" Middle = Cols.SubString(6, 6)

'النتيجة ستكون = "Second"	بأن الحرف الأول يعتبر رقم صفر في النص.		
Dim Name As String Dim Start As Short Name = "Abraham" Start = Name.IndexOf("h") 'النتيجة ستكون = 4	تحديد رقم حرف معين في النص.	Instr	IndexOf
Dim Spacey, Trimmed As String Spacey = " Hello " Trimmed = Spacey.Trim 'النتيجة ستكون = "Hello"	تحذف المسافات الفارغة، والبادئات الفارغة من النص.	Trim	Trim
Dim RawStr, CleanStr As String RawStr = "Hello333 there" CleanStr = RawStr.Remove(5, 3('النتيجة ستكون = "Hello there"	تحذف حروف من نص معين.		Remove
Dim Oldstr, Newstr As String Oldstr = "Hi Felix" Newstr = Oldstr.Insert(3, "there(" 'Newstr = "Hi there Felix"	تضيف حرف إلى نص معين.		Insert
Dim str1 As String = "Soccer" Dim str2 As String = "SOCCER" Dim Match As Short Match = StrComp(str1, _ str2, CompareMethod.Text) 'النتيجة ستكون = 0 [strings match]	تقوم بالمقارنة بين نصين، بغض النظر عن حالة الأحرف.		StrComp

فهرسة النصوص:

قبل أن يقوم الفيجوال بيسك بفهرسة النصوص يقوم أولاً بمقارنة النص حرف حرف مع قائمة تحتوي على قيمة رقمية لكل حرف، ثم يقوم بعدها بإظهار النتائج، هذه القائمة تسمى ASCII أو ANSI ، تحتوي هذه القائمة على قيمة رقمية لكل حرف وتميز بين الحرف الكبيرة والصغيرة، تبدأ القائمة من صفر حتى ٢٥٥، تمثل الحروف العربية منها من الرقم ١٩٢ أو ١٩٣، وللعلم هذه القائمة تميز بين الصيغ المختلفة للحرف الواحد في اللغة العربية فــ "أ" يختلف عن "ا" ويختلف عن "آ" وعن "إ". وعند إعطاء كل حرف من الحروف قيمة رقمية يقوم البرنامج بمقارنة القيم الرقمية ثم القيام بعملية الفهرسة. فمثلاً الحرف a بالإنجليزية يساوي القيمة ٩٧ بينما الحرف A يساوي القيمة ٦٥ وعلية فــ A يعتبر قبل a في عملية الفهرسة. قائمة الــ ASCII كانت تحتوي على ١٢٨ عنصر وفي الثمانينات من القرن الماضي أضافت شركة IBM العديد من الحروف اللاتينية والرموز حتى صارت القائمة تحتوي على ٢٥٥ بند، لكن مع دخول البرامج الكمبيوترية مرحلة العالمية بقت القائمة ASCII لا تكفي لمقارنة الحروف فكان لابد من بديل عالمي، فظهر مصطلح الــ Unicode كبديل للقائمة، تعتبر قائمة الــ Unicode قائمة معيارية عالمية تحتوي على ٦٥٥٣٦ بند وتحفظ كل الحروف العالمية، ويندوز XP و ويندوز 2003 وما بعده من نظم التشغيل وكذلك فيجوال ستديو ٢٠٠٨ تتعامل مع قائمة الــ ASCII وقائمة الــ Unicode بكل أريحية.

عند تحويل برنامجك إلى برنامج عالمي، لابد أن تتعرف أكثر على القائمة المعيارية العالمية Unicode.

التعامل مع قائمة الــ ASCII

لمعرفة الرقم المقابل لكل حرف من الحروف في قائمة الــ ASCII نستخدم الدالة ASC فالكود التالي يرجع لنا برقم الحرف "z" من القائمة:

```
Dim AscCode As Short
AscCode = Asc("z")
MsgBox (AscCode)
```

ويمكننا كتابة أية حرف بداخل علامة التنصيص لمعرفة قيمته الرقمية من القائمة ASCII، للقيام بالعملية العكسية، وهي معرفة الحرف الذي يمثل قيمة رقمية معينة نستخدم الدالة Chr كالتالي:

```
Dim letter As Char
letter = Chr(122)
MsgBox (letter)
```

ونكتب أي رقم من ٠ إلى ٢٥٥.

لمعرفة تفاصيل أكثر عن القائمة ASCII ارجع للتعليمات المرفقة مع برنامج الفيجوال بيسك وابحث عن Chr, ChrW functions وفي نهاية المقالة ستجد جول يوضع كل قيمة والحرف المقابل لها.

نعود الآن لعملية مقارنة الحروف، عند مقارنة الحروف يقوم الفيجوال بيسك باستخدام أحد المعاملات التالية <> لا يساوي، = يساوي، < أصغر من، > أكبر من، =< أصغر من أو يساوي، => أكبر من أو يساوي، فـــ "A" يعتبر أصغر من "B" لان "B" > "A". وعند مقارنة كلمة مع أُخرى يقوم الفيجوال بمقارنة الحرف الأول من الكلمة الأولى مع الحرف الأول من الكلمة الثانية ثم يقوم بمقارنة الحرف الثاني من الكلمة الأولى مع الحرف الثاني من الكلمة الثانية، حتى يصل إلى الفرق، فإذا كان لدينا الكلمتين، Michael و Mike يقوم البرنامج بمقارنة الحرف الأول ثم الثاني ثم الثالث ليعرف بأن Mike < Michael وعلية يقوم البرنامج بترتيب Michael قبل Mike في قائمة الفهرسة. إذا كانت لدينا قيمتين متساويتين وكانت إحدى القيمتين أطول من الأخرى فإن القيمة الأول تعتبر أكبر من القيمة الأقصر، فمثلاً AAAA تعتبر أكبر من AAA.

ترتيب النصوص بداخل صندوق النص:

في هذا المثال سنتعلم كيف نستخدم بعض الدوال التي تساعدنا في ترتيب النصوص بداخل صندوق النص، سنستخدم المثال السابق، الموجود في المرفق رقم ٠٣٨، مع بعض التعديلات، سنضيف إلى القوائم المنسدلة، "فتح" لفتح ملف نصي، وكذلك "إغلاق" لإغلاق الملف النصي، ونضيف "ترتيب النص"، ليقوم البرنامج بترتيب النص بداخل صندوق النص.

لترتيب النص نستخدم الإجراء ShellSort لكن البرنامج يتعامل الآن مع النص في صندوق النص كوحدة واحدة، لابد من تقطيع النص إلى مجموعة وحدات نصية ثم ترتيبها بـــ ShellSort. سيتم تقطيع النص إلى اسطر وسيقوم البرنامج بترتيب النص سطر سطر، بحث ينظر البرنامج إلى كل سطر كأنه وحدة واحدة، فإذا كان النص يحتوي على كلمة واحدة فسيقوم البرنامج بترتيب النص بشكل تصاعدي، من أول حرف حتى آخر حرف، أما إذا كان كل سطر عبارة عن أكثر من كلمة فيقوم البرنامج باعتماد الحرف الأول من كل سطر ويرتب على أساسه. لقد قام مؤلف الكتاب بكتابة وحده برمجية Module تقوم بفصل النص إلى مجموعة من الأسطر بحيث تعتمد كل سطر وحدة نصية واحدة، وقام المؤلف بإضافة الإجراء ShellSort إلى الحدة البرمجية Module لنقوم باستدعائها من أية مكان في الكود. المهم في هذا الكود هو الطريقة التي تم تقسيم النص إلى أسطر، حيث أنه لا توجد دالة في الفيجوال بيسك تحسب لنا عدد الأسطر أو تقوم بفصل النص إلى عدة أسطر، ولكن تم تصميم إجراء يقوم بالبحث بداخل النص عن الأمر Enter (حيث من المعروف بأننا إذا أردنا أن نضيف سطراً جديداً في النص، نقوم بالضغط على الزر Enter في الكيبورد)، وإذا رجعنا لقائمة معايير الـــ ASCII لوجدنا بأن الرقم المقابل لـــ Enter في القائمة يساوي ١٣، فالإجراء الذي يحدد السطر الجديد يقوم بالبحث بواسطة الدالة Chr على الرمز ١٣ في النص بداخل صندوق النص، وعليه يتم تحديد كل سطر جديد، راجع الكود التالي الذي يقوم بتحديد كل سطر جديد في صندوق النص:

```
Dim ln, curline, letter As String
Dim i, charsInFile, lineCount As Short
'determine number of lines in text box object (txtNote)
lineCount = 0 'هذا المتغير يقوم بحساب عدد الأسطر
charsInFile = txtNote.Text.Length 'الحصول على إجمال عدد الأحرف في النص
For i = 0 To charsInFile - 1 'التحرك بداخل النص حرف حرف
letter = txtNote.Text.Substring(i, 1) 'أخد كل حرف على حده
If letter = Chr(13) Then ' إذا كان الحرف Enter
```

```
lineCount += 1 ‘(أضف إلى عدد الأسطر ١) إذهب للسطر الذي يليه
i += 1 'skip linefeed char (typically follows cr on PC)
End If
Next i
```

في الكود أعلاه تم إسناد عدد الأسطر إلى المتغير lineCount، وسنستفيد من المتغير لتعريف مصفوفة متغيرة ونحدد عدد عناصرها بعدد الأسطر. وبعد تعبئة الأسطر في المصفوفة المتغيرة، نستخدم الدالة Sort في المصفوفة لترتيب الأسطر ثن نقوم بعرض الأسطر في صندوق النص. لنجرب الآن تشغيل التطبيق (بالمرفق رقم ٠٣٩) ونكتب الأسماء التالية في صندوق النص:

Zebra
Gorilla
Moon
Banana
Apple
Turtle
تمر
حصان
نخلة

لا بد أن نتعمد عدم ترتيب العناصر أعلاه ليقوم التطبيق بترتيب العناصر المذكورة، ولابد أن نتأكد أننا ضغطنا على Enter بعد كتابة الكلمة الأخيرة في القائمة "نخلة" ليقوم البرنامج بحساب عدد الأسطر بشكل صحيح. بعد إضافة الأسماء إلى صندوق النص، نختار ترتيب من القائمة المنسدلة، ليقوم البرنامج بترتيب الأسماء كاملة بشكل ألفبائي وبحيث يبدأ بالكلمات الإنجليزية ثم يقوم بترتيب الكلمات العربية (لان ترتيب الحروف الإنجليزية في قائمة ASCII يبدأ من ٣٢ وينتهي في ١٢٧، أم الحروف العربية فتبدأ عند ترتيب ١٩٢ وما بعد ذلك). بعد ترتيب النص ستكون الفورم كما في الشكل التالي:



يتبقى لنا هنا ملاحظتين على عملية ترتيب النصوص، الأولى: إذا كان لدينا نصوص وأرقام في نص معين وأردنا الترتيب سيقوم البرنامج بترتيب الأرقام أولاً ثم يقوم بترتيب النصوص بعدها، لان الأرقام تسبق النصوص في قائمة ASCII، الملاحظة الثانية لا يُنصح باستخدام قائمة الـــ ASCII في ترتيب الأرقام لإن قائمة الـــ ASCII تتعامل مع الأرقام كنصوص، فإذا كان لدينا الأرقام من صفر إلى عشرة في قائمة فسيقوم التطبيق بوضع الصفر في أول القائمة ثم الواحد ثم العشرة ثم الإثنين، لإنه سيتعامل مع الـــ ١٠ وكأنها نص (كأنها حرفين مستقلين)، فسيبدأ بالتعامل مع الرقم ١ لإنه في أول السطر ثم سيذهب للتعامل مع العنصر التالي الـــ ٢ بغض النظر عن الصفر الذي يتبع الرقم واحد، ولحل هذه المشكلة يجب ترتيب الأرقام بشكل رقمي وليس بشكل نصي، وفي حالة معالجة الأرقام بشكل نصي يجب أن نمنع المستخدم من إدخال نصوص غير رقمية إلى مربع النص.

خطوة إضافية للأمام: مراجعة كود تطبيق ترتيب النصوص

تطبيق ترتيب النصوص (بالمرفق ٠٣٩) يحتوي على العديد من الدوال والإجراءات التي يجب أن نعرفها لنزداد معرفة برمجية، نفتح التطبيق ثم نذهب إلى منطقة الكود التابعة للنموذج Form1 ونذهب للحدث Click التابع لزر القائمة المنسدلة SortTextToolStripMenuItem، قد تناولنا النصف الأول من هذا الكود والذي يقوم بتقسيم النص إلى مجموعة من الأسطر، لنذهب إلى الشطر الثاني من الكود والذي يقوم بإعادة تعريف المصفوفة المتغيرة strArray التي سبق

وعرفناها في الوحدة البرمجية Module1، قمنا هنا بإعادة التعريف لتحديد عدد عناصر المصفوفة والذي يساوي عدد الأسطر في النص. ثم قمنا بتعبئة المصفوفة بعناصر الأسطر الموجودة في النص، بعد ذلك قمنا بترتيب هذه العناصر باستخدام الإجراء ShellSort الذي عرفناه في الوحدة البرمجية Module ، ثم قمنا في آخر الكود بإدخال عناصر المصفوفة (بعد الترتيب) إلى صندوق النص.

بالنسبة للحدة البرمجية الـــ Module قمنا بتعريف المصفوفة المتغيرة strArray لنقوم باستخدامها في مكان في الكود على طول التطبيق، كما قمنا بتعريف الإجراء ShellSort والذي يقوم بترتيب عناصر المصفوفة على أساس تقسيم عناصر المصفوفة إلى قائمتين مصغرتين كل قائمة تحتوي على نصف العناصر ثم نقوم بالترتيب، الترتيب النهائي يكون ترتيب تصاعدي أما إذا أردنا ترتيب تصاعدي فتقوم بقلب الرمز "=>" إلى "=<".

بقية الكود الخاص بفتح الملفات النصية وكذلك حفظ الملفات النصية وكود الخروج من البرنامج أو إغلاق الملف النصي قد تعرفنا عليها في الأمثلة السابقة.

وصلنا الآن إلى نهاية الفصل الثالث عشر، وهذا يعني أننا انتهينا من الجزء الثاني من الكتاب (أساسيات البرمجة) لندخل في الجزء الثالث من الكتاب والذي يعنى بواجهة المستخدم وطُرق تحسينها.

خلاصة الفصل الثالث عشر

من اجل أن	**قم بالتالي**
فتح ملف نصي	نستخدم الدالة FileOpen لفتح الملفات النصية مثال: FileOpen(1, OpenFileDialog1.FileName, _ OpenMode.Input)
الحصول على سطر من الملف النصي	نستخدم الدالة LineInput مثال Dim LineOfText As String

	LineOfText = LineInput(1)
الذهاب إلى نهاية الملف النصي	نستخدم الدالة EOF: مثال Dim LineOfText, AllText As String Do Until EOF(1) LineOfText = LineInput(1) AllText = AllText & LineOfText & _ vbCrLf Loop
إغلاق ملف نصي	نستخدم الدالة FileClose، مثال FileClose(1)
عرض ملف نصي باستخدام LineInput	نستخدم الدالة LineInput لأخذ النص من الملف النصي إلى متغير نصي، ثم نقوم بتفريغ قيمة المتغير النصي بداخل صندوق نص، مثال: Dim AllText, LineOfText As String Do Until EOF(1) 'قراءة الأسطر من الملف LineOfText = LineInput(1) AllText = AllText & LineOfText & vbCrLf Loop txtNote.Text = AllText 'عرض الملف
عرض ملف نصي باستخدام الفئة	نضيف الكود Imports System.IO أعلى منطقة الكود في النموذج (الفورم) ثم نستخدم الفئة StreamReader لعرض محتويات ملف نصي

StreamReader	بداخل صندوق نص، مثال: Dim StreamToDisplay As StreamReader StreamToDisplay = New StreamReader(_ "c:\vb08sbs\chap13\text browser\badbills.txt") TextBox1.Text = StreamToDisplay.ReadToEnd StreamToDisplay.Close() TextBox1.Select(0, 0)
عرض ملف نصي باستخدام مجال الأسماء My	نستخدم مجال الأسماء My.Computer.FileSystem وكذلك الطريقة ReadAllText ولنفرض بأننا نريد أن نعرض الملف في صندوق النص txtNote كالتالي: Dim AllText As String = "" AllText = _ My.Computer.FileSystem.ReadAllText("C:\txtfile.txt") txtNote.Text = AllText 'عرض الملف
لعرض نافذة فتح ملف	نضيف الأداة OpenFileDialog ثم نظهر النافذة بالكود: OpenFileDialog1.ShowDialog()
لعرض نافذة حفظ ملف	نضيف الأداة SaveFileDialog ثم نقوم بإظهارها بواسطة الكود كالتالي: SaveFileDialog1.ShowDialog()
إنشاء ملف نصي جديد	نستخدم الدالة FileOpen لإنشاء ملف نصي كالتالي: FileOpen(1, SaveFileDialog1.FileName, _

	OpenMode.Output)
لحفظ النص إلى ملف نصي	نستخدم الدالة PrintLine كالتالي: PrintLine(1, txtNote.Text)
تحويل حرف إلى القيمة ASCII	نستخدم الدالة Asc كالتالي: Dim Code As Short Code = Asc("A") 'الناتج سيكون 65
تحويل قيمة الــ ASCII إلى حرف	مستخدم الدالة Chr كالتالي: Dim Letter As Char Letter = Chr(65) 'الناتج سيكون "A"
استخراج نص من داخل نص آخر	نستخدم الطريقة Substring التابعة لإطارات العمل دوت نت أو نستخدم الدالة Mid التابعة للفيجوال بيسك مثال: Dim Cols, Middle As String Cols = "الأول الثاني الثالث" Middle = Cols.SubString(6, 6) 'النتيجة ستكون "الثاني"

الجزء الثالث: تصميم وتحسين واجهة المستخدم

الفصل الرابع عشر : التعامل مع النماذج (Forms)، وأدوات التحكم (Controls)، وقت تشغيل البرنامج

إضافة نماذج جديدة إلى التطبيق

حتى الآن قمنا بتصميم العديد من التطبيقات، لكن كلها كانت بنموذج (فورم) واحد مع نوافذ حوار،وصناديق حوار، ونوافذ لفتح ولحفظ الملفات، كل الاستخدامات السابقة للنماذج وكذلك لصناديق الحوار كانت للحوار مع المستخدم فقط، ولكن ماذا إذا أدرنا التعامل بشكل متقدم مع المستخدم، كعرض بيانات أكثر وبشكل منظم، لابد من إضافة نماذج جديدة للمشروع، كل نموذج جديد نقوم بإضافته يقوم بوراثة قدراته من الفئة System.Windows.Forms.Form. يقوم الفيجوال بيسك ٢٠٠٨ بتسمية النماذج، فأول نموذج يسمى Form1 والثاني Form2 وهكذا ونستطيع تغيير اسم النموذج من نافذة الخصائص عند الخاصية Name أو عند نافذة حوار إضافة نموذج جديد، كل نموذج جديد لابد أن يحتوي على اسم فريد وأدوات خاصة به، وخصائص وكذلك إجراءات خاصة.

كيف نستخدم النماذج

يعطينا الفيجوال بيسك الخيار في التعامل مع النماذج، فبإمكاننا عرض جميع النماذج في وقت واحد وبإمكاننا عرض كل نموذج في وقت الحاجة إليه، ونستطيع عرض أكثر من نموذج لكن نتحكم في المستخدم بحيث يستخدم نموذج معين أولاً دون غيره ثم نسمح له باستخدام نموذج معين أو بقية النماذج. إذا كان لدينا نموذجين أردنا فتح نموذج معين للمستخدم و لا نريد أن نسمح له باستخدام النموذج الآخر فيمكننا استخدام ShowDialog أما إذا أردنا السماح للمستخدم باستعمال النموذجين في نفس الوقت فنستخدم Show.

التعامل مع أكثر من فورم

هل تتذكر برنامج Arqam الذي صممناه في بداية الكتاب (في المرفق ٠٠٢)، ثم قمنا بالتعديل عليه في الفصل العاشر (بالمرفق ٠٢٩) لعلك تذكرت، لنأخذ البرنامج في المرفق ٠٢٩ ثم نحاول التعديل عليه بإضافة نموذج جديد للبرنامج ليقوم هذا النموذج بعرض معلومات عن التطبيق، سنسمي هذا النموذج HelpInfo، بعد فتح البرنامج نذهب إلى أعلى يمين بيئة التطوير حيث اسم البرنامج Arqam ثم نضغط Right-Click عليه، ونختار Add ومنها New Item سيفتح لنا الفيجوال بيسك ٢٠٠٨ نافذة حوار من خلالها نختار ما نريد أن نضيف للمشروع، بإمكاننا إضافة نموذج (فورم)، فئة (كلاس)، أو شاشة حول، أو أداة خاصة أو أداة تقارير أو غيرها، نختار هنا

Windows Form ونقوم بتسميتها HelpInfo.vb في نفس النافذة، سيقوم الفيجوال بيسك بإضافتها إلى المشروع.

ملاحظة: هناك العديد من النماذج التي قد نحتاجها لإضافتها للمشروع، وتلك النماذج مصممة بشكل شبه جاهز للعمل عليها، مثل Explorer Form والذي يضيف لنا مستكشف شبيه بمستكشف الويندوز، أو يمكن إضافة شاشة "حول البرنامج"، معرفة هذه النماذج تسهل لنا عملية إضافتها للمشروع في وقت الحاجة، بإمكانك إضافة ما تريد لمشروعك وتجربته ثم حذفه، لحذف ما أضفته لمشروعك قم بوضع الماوس على النموذج ثم اختر Exclude from Project فسيقوم الفيجوال بيسك بحذف النموذج من المشروع بدون أن يحذفه من جهاز الكمبيوتر، أما إذا أردت أن تحذف نموذج من المشروع وجهاز الكمبيوتر فاختر Delete.

بعد إضافة النموذج HelpInfo إلى مشروعك قم بفتحه ثم أضف إليه بعض الكائنات وقم بتعديل بعض خصائصه ليصبح النموذج كالتالي بعد التعديل عليه:



الآن نضيف الكود التالي للحدث Click التابع للزر "موافق":

```
Me.DialogResult = DialogResult.OK
```

لتوضيح الكود أعلاه يجب علينا أولاً بأننا سنتعامل مع النموذج HelpInfo على أساس أنه نموذج للحوار (مثل صناديق الحوار)، عند التعامل مع النموذج على هذا الأساس يجب أن نكتب في الكود

الذي يقوم بإظهار النموذج التالي: ShowDialog وليس Show، ففي حالة كتبنا ShowDialog في أمر إظهار النموذج فسيظهر النموذج على أساس أنه نافذة حوار وبسبب ذلك لن يستطيع المستخدم الرجوع إلى النموذج الأولي إلا بعد أن يغلق النموذج الحواري، أما إذا استخدمنا الأمر Show لوحدها فسيتعامل النموذج مع المستخدم على أساس أنه نموذج مستقل، ويستطيع المستخدم التعامل مع النموذج المفتوح والنموذج الأولي في نفس الوقت. ثانياً بعد فتح النموذج على أساس لأنه نموذج للحوار لابد أن يعود هذا النموذج مع نتيجة الحوار، والنتيجة ستكون بالنفي أو بالإيجاب، فالكود أعلاه يعني أن نتيجة الحوار بالإيجاب، ونستطيع وضع أكثر من زر أحدهما "موافق" والآخر "غير موافق" والثالث "إلغاء الأمر" وكتابة الكود لكل زر من الأزرار، حيث ستتغير الكلمة الأخيرة مع كل زر.

بعد فتح النموذج سيقوم هذا النموذج بفتح ملف نصي في التعليمات ليقرأه المستخدم، وسنستخدم الفئة StreamReader ولذلك لابد من إضافة مجال الأسماء التالي في أول الكود:

```
Imports System.IO
```

استيراد مجال الأسماء System.IO سيسهل لنا التعامل مع الفئة StreamReader، نقوم بعد ذلك بكتابة الكود التالي في الحدث Load التابع للنموذج ليقوم النموذج بعرض معلومات الملف النصي بعد فتحه مباشرة على افتراض أن الملف النصي في المسار \:C أما إذا كان الملف النصي في مكان آخر فيمكننا تغيير المسار في مرحلة الكود:

```
Dim StreamToDisplay As StreamReader
StreamToDisplay = _
New StreamReader("c: \readme.txt")
TextBox1.Text = StreamToDisplay.ReadToEnd
StreamToDisplay.Close()
TextBox1.Select(0, 0)
```

في الكود أعلاه قمنا بتعريف المتغير StreamToDisplay على أساس أنه StreamReader ثم سمحنا لـــ StreamToDisplay بقراءة ملف نصي في المسار \:C، بعدها قمنا بإسناد النص

الموجود في المتغير StreamToDisplay إلى صندوق النص TextBox1 ليقوم بعرض النص الموجود في الملف النصي بداخل صندوق النص، بعدها قمنا بإغلاق الملف النصي (بالطبع بعد أن تكون محتوياته على صندوق النص في النموذج)، وإذا كان هناك تظليل في النص الموجود في صندوق النص قمنا بإزالة ذلك التظليل بواسطة السطر الأخير من الكود.

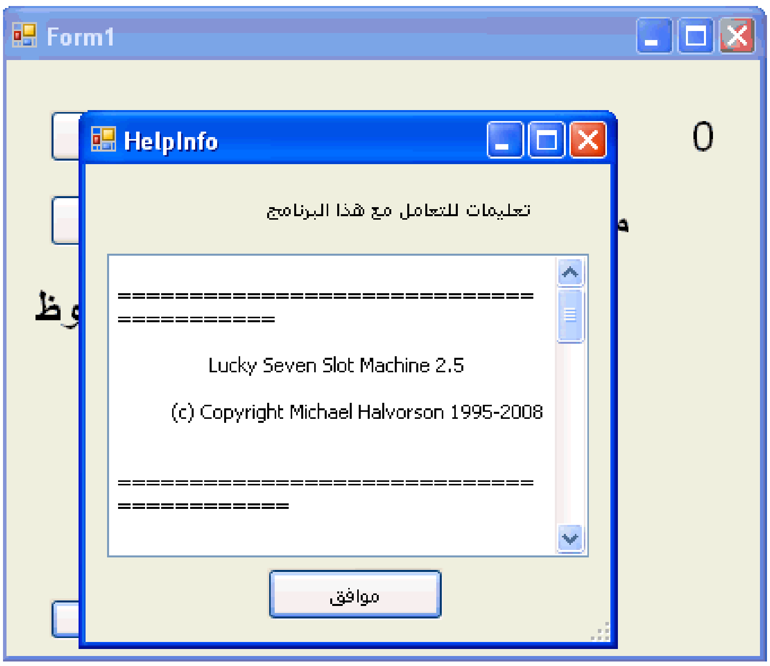
الآن انتهينا من نموذج HelpInfo وسنعود إلى الفورم الأولى لكتابة كود يسمح للنموذج HelpInfo بالعرض، ولتنفيذ ذلك نضيف زر على النموذج الأول ونعدل الخاصية Text له على "معلومات"، نقوم بالضغط Double-Click على الزر المُضاف ونكتب الكود التالي:

```
My.Forms.HelpInfo.ShowDialog()
```

ليقوم التطبيق بإظهار الفورم HelpInfo على أساس أنها نافذة للحوار. للعلم نستطيع كتابة الكود أعلاه بطريقة أُخرى كالتالي:

```
HelpInfo.ShowDialog()
```

نقوم الآن بتنفيذ البرنامج، ونتأكد من وجود الملف النصي readme على المسار \:C ثم نقوم بالنقر على الزر "معلومات" من النافذة التي تظهر وستظهر لنا النموذج الجديد الذي أضفناه مع المعلومات الموجودة في الملف النصي المذكور في المسار المحدد كالتالي:



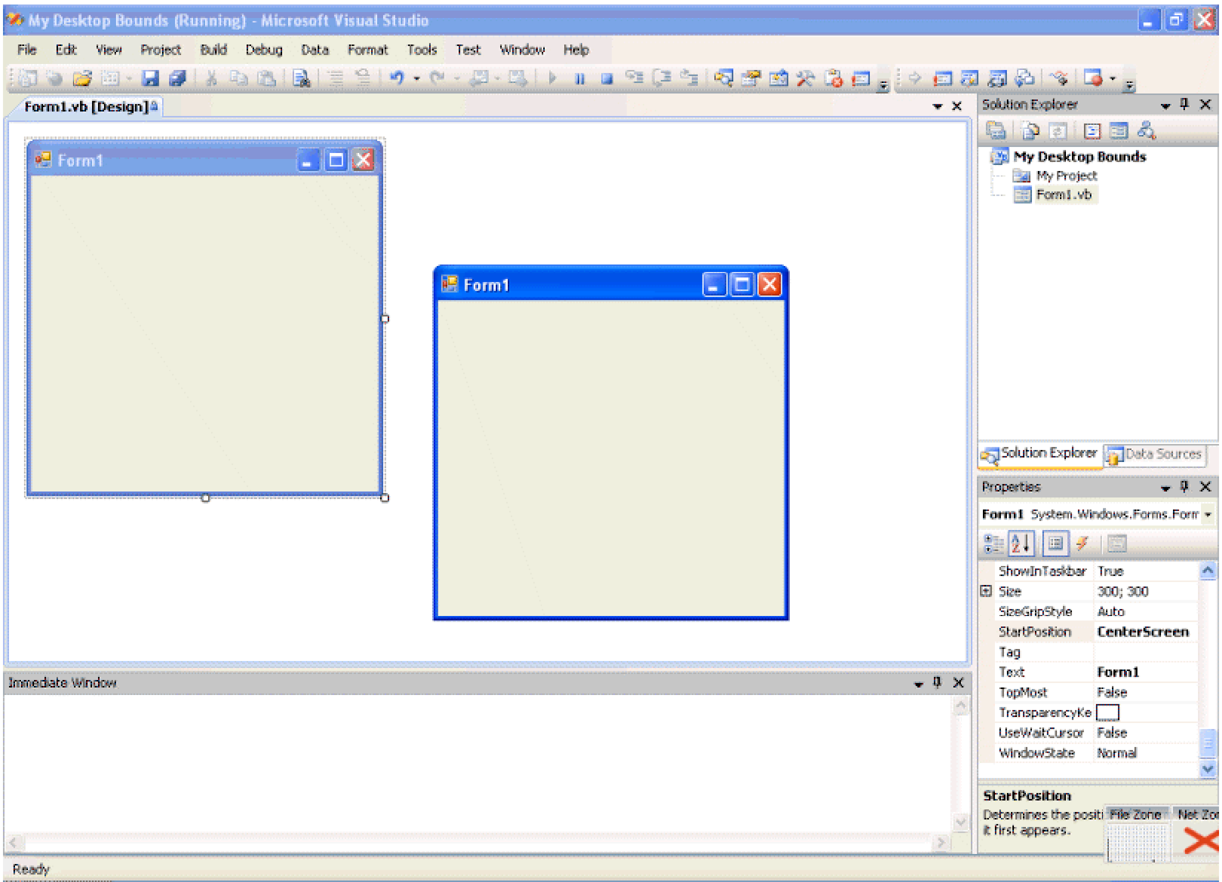
المشروع موجود في المرفق رقم ٠٤٠.

تحديد موقع النماذج على سطح المكتب:

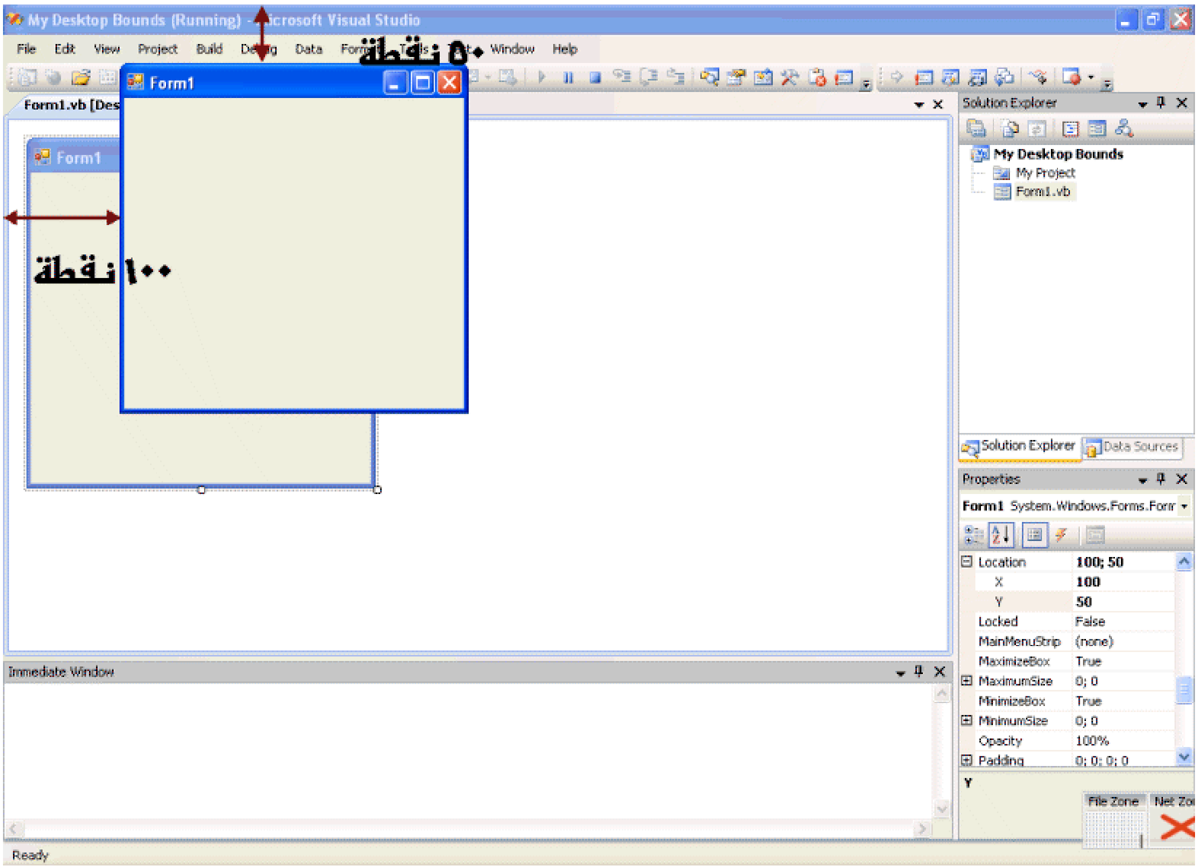
بعد تصميم البرنامج الذي قد يحتوي على أكثر من نموذج، نحتاج لأن يظهر النموذج الأول في وسط الشاشة، ونموذج آخر في أعلى الشاشة والنموذج الثالث في وسط النموذج الثاني وهكذا، أو قد نحتاج لتغيير مكان ظهور نموذج معين إذا اختار المستخدم خيار معين. في الفيجوال بيسك ٦ وبعد إضافة النموذج للمشروع يُظهر لنا الفيجوال بيسك خانة صغيرة تسمى **Form Layout** فيها شاشة النموذج بداخل شاشة الويندوز وباستخدام الماوس نستطيع تغيير مكان ظهور النموذج في وقت تشغيل البرنامج. أما في فيجوال بيسك ٢٠٠٨ فالأمر يتغير لا وجود للـــ **Form Layout** وإنما نستطيع تغيير مكان النموذج على الشاشة بواسطة خاصية تسمى **DesktopBounds**. في نفس الوقت نستطيع تحديد مكان النموذج باستخدام الخاصية **StartPosition** ومنها نستطيع أن نختار أحد هذه الخيارات: **Manual** وعند تحديد هذا الخيار لابد أن نذهب إلى الخاصية **Location** ونقوم بكتابة مكان ظهور النموذج بالأرقام، **CenterScreen** يقوم بوضع النموذج في وسط الشاشة وهذا الخيار مفضل عند كثير من المبرمجين، **WindowsDefaultLocation** هذا الخيار الطبيعي المعتمد في حالة لم تقم بتحديد أحد الخيارات ويقوم بوضع النموذج في المكان الذي يراه نظام التشغيل مناسباً وعادة ما يكون النموذج في أعلى يسار الشاشة، **WindowsDefaultBounds** هذا الخيار يقوم أولاً بتغيير طول وعرض النموذج ثم يقوم بوضعه في المكان الذي يقترحه نظام الويندوز وعادة ما يكون أعلى يسار الشاشة، أما الخيار **CenterParent** فيعتبر مناسب للتطبيقات الكبيرة التي تحتوي على أكثر من نموذج (النموذج الأب والنماذج الأبناء) ويقوم بوضع النموذج الابن في وسط شاشة النموذج الأب (فمثلاً برنامج المبيعات والمخازن الرئيسي يعتبر النموذج الأب، أما شاشة إصدار الفواتير فتعتبر النموذج الابن ومن الأفضل وضع النموذج الابن في وسط شاشة النموذج الأب). المثال التالي يوضح كيفية وضع النموذج في المكان المناسب على الشاشة:

١-نقوم بإنشاء برنامج باسم **My Desktop Bounds**

٢-بعد ظهور النموذج Form1 نذهب إلى خصائص النموذج ونختار منها الخاصية StartPosition ونضبطها على الخيار CenterScreen نقوم بتنفيذ البرنامج ماذا سنلاحظ:



نلاحظ بأن النموذج يظهر في منتصف الشاشة بالضبط، نغلق البرنامج ونذهب إلى الخاصية StartPosition ومنها نختار Manual ، بعد تحديد هذا الخيار سيعتمد الفيجوال بيسك على الخاصية Location التابعة للنموذج ليحدد مكان النموذج، وعليه نذهب للخاصية Location ونكتب القيم (١٠٠، ٥٠) فيها ثم نقوم بتنفيذ البرنامج، سيقوم الفيجوال بيسك بتحريك النموذج بمقدار ١٠٠ نقطة من يسار الشاشة وبمقدار ٥٠ نقطة من أعلى الشاشة: القياسات هنا بالبكسل (ونسميه نقطة):



تعلمنا الآن استخدام الخاصية StartPosition والآن سنتعلم كيفية التعامل مع النموذج الثاني في مشروعنا باستخدام DesktopBounds سنقوم الآن بإضافة نموذج جديد للمشروع بواسطة الكود (لن نستخدم Add New Item) وستقوم بتحديد مكان هذا النموذج على الشاشة بواسطة الكود كذلك كالتالي:

١- نقوم بإضافة زر جديد للنموذج الحالي، وفي خاصية Text نكتب "إضافة نموذج"

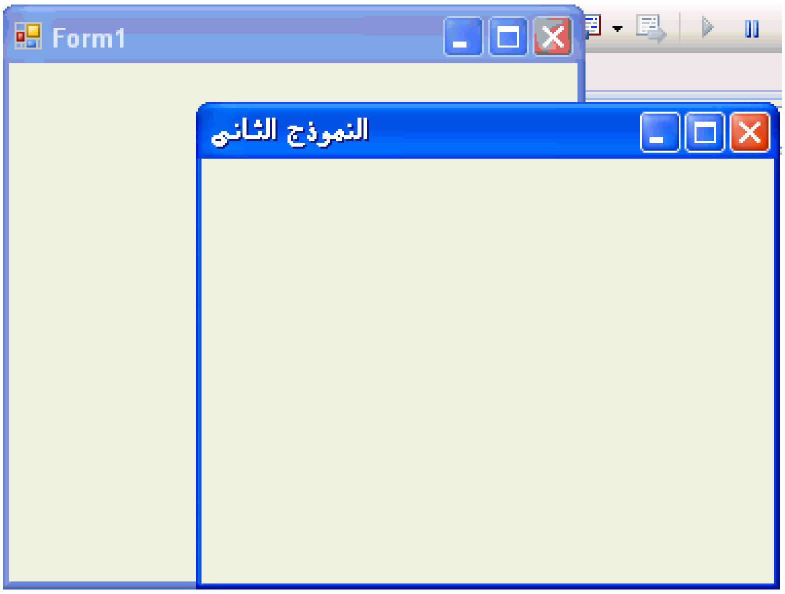
٢- نقوم بالضغط نقرتين على الزر وفي حدث Click التابع للزر نكتب هذا الكود:

```
'إضافة نموذج جديد للمشروع باسم Form2
Dim form2 As New Form
'نقوم بتحديد الخاصية Text التابعة للنموذج الجديد بالنموذج الثاني
' في نفس الوقت نقوم بتحديد شكل النموذج
form2.Text = "النموذج الثاني"
form2.FormBorderStyle = FormBorderStyle.FixedDialog
'نقوم بتغيير الخاصية StartPosition إلى Manual
form2.StartPosition = FormStartPosition.Manual
'نقوم بتعريف متغير مستطيل لحفظ مقاسات النموذج
'المسافات من أعلى واليسار هي (200, 100)
'العرض والإرتفاع للنموذج هما (300, 250)
Dim Form2Rect As New Rectangle(200, 100, 300, 250)
```

```
'الثاني النموذج إلى أعلاه المستطيل إحداثيات بنقل نقوم
form2.DesktopBounds = Form2Rect
'حواري نموذج أنه أساس على الجديد النموذج بعرض نقوم
form2.ShowDialog()
```

لاحظ الفرق بين التعليقات والكود حيث أن التعليقات تبدأ بـــ ' ولونها أخضر

الكود أعلاه لا يحتاج إلى شرح وإنما يحتاج لقراءة التعليقات، وللعلم توجد نسخة من المشروع أعلاه في المرفق ٠٤١، نقوم الآن بتنفيذ البرنامج ونقوم بالضغط على الزر "إضافة نموذج" يا إلهي ماذا يحدث، لقد ظهر لنا نموذج جديد (الذي قمنا بتصميمه برمجياً فقط) مع الخصائص التي حددناها، لاحظ الصورة:



تصغير وتكبير واستعادة النماذج

نستطيع تكبير وتصغير نموذج معين بواسطة الكود، قبل ذلك لابد أن نتأكد من وجود زر التصغير وزر التكبير في أعلى النموذج وإذا لم فلابد من تفعيلهما من الكود كالتالي:

```
form2.MaximizeBox = True
form2.MinimizeBox = True
```

بعدها نستطيع تكبير وتصغير النموذج واستعادته بالكود كالتالي، كود التصغير:

```
form2.WindowState = FormWindowState.Minimized
```

كود التكبير :

```
form2.WindowState = FormWindowState.Maximized
```

كود الاستعادة:

```
form2.WindowState = FormWindowState.Normal
```

لحصر تكبير أو تصغير النموذج إلى مقاسات معينه بحيث لا يستطيع المستخدم تكبير النموذج أكبر من تلك المقاسات، و لا يستطيع المستخدم التصغير أقل منها، نستطيع ذلك من الخاصية MaximumSize والخاصية Minimumize، ونستطيع تحديد ذلك بواسطة الكود بحيث نعرف قالب للمساحة ثم نسحب مساحة هذا القالب إلى الخاصية MaximumSize أو للخاصية Minimumize كالتالي:

```
Dim FormSize As New Size(400, 300)
form2.MaximumSize = FormSize
```

إضافة الكائنات إلى النماذج وقت تشغيل البرنامج

في المثال السابق (والموجود في المرفق ٠٤١) قمنا بتعريف نموذج جديد Form2 بواسطة الكود وقمنا كذلك بإضافته للمشروع، ليس ذلك فقط فقد قمنا بتحديد العديد من الخصائص التابعة له بواسطة الكود، تلك العملية حفزتنا لنقوم بالعديد من المهام بواسطة الكود، فمثلاً نستطيع إضافة الكائنات (أزرار، صناديق نص، وغيرها) بواسطة الكود. سيفيدنا هذا كثيراً في برامجنا الكبيرة إذا تم ربط تلك العملية بقواعد البيانات فسيكون برنامجنا متقدماً ومثالياً وذو خمسة نجوم. نستطيع إضافة الكائنات بواسطة الكود بتعريف الكائنات أولاً ثم إضافتها للنماذج فلإضافة زر نقوم أولاً بتعريف الزر:

```
Dim button1 As New Button
```

طبعاً قبل التعريف أعلاه يجب أن نتأكد من عدم وجود زر سابق يحتوي على نفس الاسم button1، بعد ذلك يمكننا تحديد خصائص الكائن المُضاف إلى النموذج كالتالي:

```
button1.Text = "Click Me"
```

```
button1.Location = New Point(20, 25)
```

الكود الأول يقوم بتعبئة الخاصية Text التابعة للزر button1 والكود الثاني يقوم بتحديد مكان الزر على النموذج. بعد تعريف الكائن وتحديد الخصائص المهمة نقوم بإضافة الكائن للنموذج:

```
form2.Controls.Add(button1)
```

قمنا في الكود أعلاه بإضافة الزر button1 إلى النموذج form2، أما في حالة ما إذا كانت الإضافة هي لنفس النموذج الذي يوجد فيه الكود فيمكننا كتابة الكود أعلاه بطريقة أُخرى:

```
Me.Controls.Add(button1)
```

فالكود أعلاه يقوم بإضافة الزر button1 إلى الفورم الحالي، في المثال التالي سنتعلم كيفية إضافة ليبل Label وكذلك زر لنموذج بواسطة الكود وكذلك كيف نجعل الليبل Label يقوم بعرض التاريخ الحالي:

١-نقوم بإنشاء مشروع جديد باسم My Add Controls

٢-نستعرض النموذج Form1، ثم نقوم بإضافة زر Button إلى النموذج ونغير الخاصية Text للزر إلى "إظهار التاريخ".

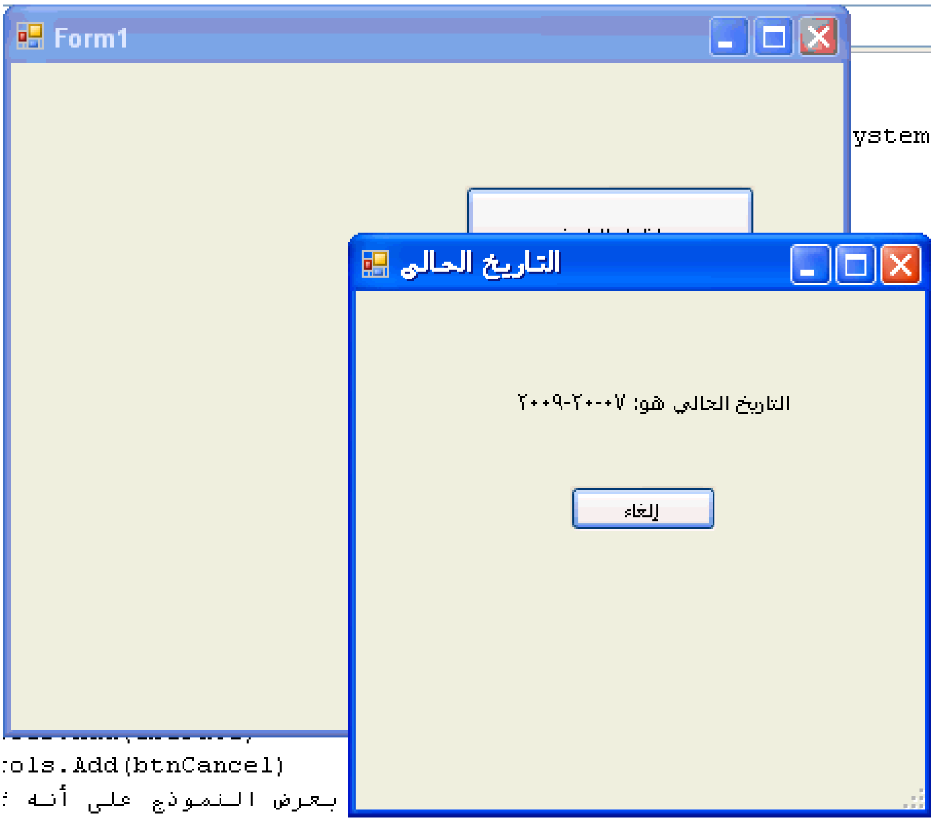
٣-نقوم بالنقر مرتين على الزر Button1 ونكتب الكود التالي في محرر الكود:

```
'نقوم بتعريف نموذج جديد مع مكوناتها
Dim form2 As New Form
Dim lblDate As New Label
Dim btnCancel As New Button
'نحدد الخصائص التابعة لليبل
lblDate.Text = "التاريخ الحالي هو: " & DateString
lblDate.Size = New Size(150, 50)
lblDate.Location = New Point(80, 50)
```

```
'وكذلك خصائص الزر
btnCancel.Text = "إلغاء"
btnCancel.Location = New Point(110, 100)
'وخصائص النموذج
form2.Text = "التاريخ الحالي"
form2.CancelButton = btnCancel
form2.StartPosition = FormStartPosition.CenterScreen
'نقوم بإضافة كائنات جديدة إلى مجموعة الكائنات على النموذج
form2.Controls.Add(lblDate)
form2.Controls.Add(btnCancel)
'نقوم بعرض النموذج على أنه نموذج حواري
form2.ShowDialog()
```

في السطور الثلاثة الأولى من الكود قمنا بتعريف النموذج الجديد، لاحظ سهولة تعريف نموذج جديد في الكود ليتم إظهاره لاحقاً وكل هذا بالكود فقط بدون إضافة حقيقية وقت التصميم هذه الميزة تسهل علينا العديد من المهام البرمجية في البرامج الكبيرة والمتوسطة، لاحظ كذلك التعليقات على الكود بين الأسطر البرمجية (السطر الرابع، الثامن، الحادي عشر، الخ) هذه التعليقات تسهل علينا عملية قراءة الكود لاحقاً، تخيل بأننا نقوم بقراءة الكود بعد ثلاث سنوات من كتابته أو تخيل بأننا نعمل كفريق على تطوير برنامج واحد فلابد أن يفهم البقية كل سطر في الكود، عند تحديد خصائص الزر و الليبل قمنا بتحديد المساحة لكل منهما وكذلك المكان الجغرافي على النموذج، أما النموذج الجديد فقد قمنا بتحديد منطقة تواجده على الشاشة حيث توسط الشاشة بالخاصية StartPosition. أما الخاصية CancelButton فتعنى بنقل الضغط على الزر Esc في الكيبورد إلى الزر إلغاء (بمعنى آخر إذا ضغطنا على الزر إلغاء أو الأمر Esc في الكيبورد

فكلاهما يقومان بنفس العمل) والذي يقوم بدوره بإغلاق النموذج. نقوم الآن بالضغط على F5 لتنفيذ البرنامج، ثم نضغط على الزر "إظهار التاريخ" سنرى مثل هذه الشاشة:



قد تلاحظ أن التاريخ في جهازي (كما في الصورة) يبدأ بالشهر ثم باليوم ثم بالسنة، نستطيع تغيير طريقة ظهور التاريخ من لوحة التحكم ثم الإعدادات الإقليمية.

التطبيق أعلاه موجود في المرفقات رقم ٠٤٢.

ترتيب الكائنات على النموذج

عند إضافة الكائنات بطريقة برمجية إلى الفورم نلاقي بعض الصعوبات في تحديد مكان وضع الكائن على النموذج (المكان الجغرافي إن صح التعبير)، حيث لابد من تحديد النقطة الأفقية التي يبدأ الكائن عندها وكذلك تحديد النقطة العمودية، راجع الكود للتطبيق السابق:

```
btnCancel.Location = New Point(110, 100)
```

فلابد من تحديد النقطتين x و y على النموذج، ولتحديد هذه النقطتين لابد من التجربة والخطأ في مرحلة التصميم وتجربة الكود، لكن هناك طُرق ملتوية لتفادي مثل هذه الصعوبة ومنها القيم التابعة بالخاصيتين Anchor و Dock حيث تحافظ الأولى على الابتعاد مسافة معينة من حافة النموذج، بينما تقوم الخاصية الثانية بإجبار الكائن بالالتصاق بإحدى حواف النموذج، نستطيع التعامل مع الخاصيتين بواسطة الكود وبهذه الطريقة نستطيع تنظيم الكائنات على الفورم إلى حد ما. في المثال التالي سنتعلم كيفية استخدام Anchor و Dock:

١-نقوم بإنشاء تطبيق جديد ونسميه My Anchor and Dock

٢-بعد فتح النموذج نقوم بإضافة صندوق للصورة Picture Box في أعلى وسط النموذج، ونضيف صندوق نص TextBox ونعدل الخاصية MultiLine على True ثم نقوم بتكبير صندوق النص ليشمل النصف السفلي من النموذج، ثم نضيف زر Button إلى أسفل يمين النموذج.

٣-نقوم بتعديل بعض خصائص النموذج وكذلك الكائنات على النحو التالي:

صندوق الصورة	Image	نختار الصورة sun.ico المرفقة
الزر Button1	Text	"ترتيب الآن"
صندوق النص	Text	"مثال على ترتيب الكائنات على النموذج"

ستكون الفورم كما في الصورة:



نضغط Double-Click على الزر "ترتيب الآن" ونكتب هذا الكود:

```
PictureBox1.Dock = DockStyle.Top
TextBox1.Anchor = AnchorStyles.Bottom Or _
AnchorStyles.Left Or AnchorStyles.Right Or _
AnchorStyles.Top
Button1.Anchor = AnchorStyles.Bottom Or _
AnchorStyles.Right
```

قمنا بتغيير الخاصية Dock لصندوق الصورة إلى DockStyle.Top ليقوم صندوق الصورة بالالتصاق بأعلى النموذج، أما صندوق النص فسيحافظ على المسافة الموجودة بينه وبين حواف النموذج الأربع، فإذا كان قريب من نهاية النموذج بمسافة معينة فسيحافظ عليها حتى إذا قمنا بتكبير الفورم أو تصغيره، وسيقوم الزر بالمحافظة على المسافة بينه وبين الحافة السفلية وكذلك اليمنى للنموذج، راجع الكود. قم الآن بتنفيذ البرنامج وقم بتغيير حجم النموذج تكبيراً وتصغير وقم بالضغط على الزر "ترتيب الآن"، قم بالضغط على الزر ثم قم بتكبير الفورم، كرر العملية مرات وقم بتغيير ترتيب الضغط على الزر ثم تعديل مساحة النموذج ولاحظ التغييرات التي تحدث للنموذج وللكائنات الموجودة عليه. المثال موجود بالمرفق ٠٤٣.

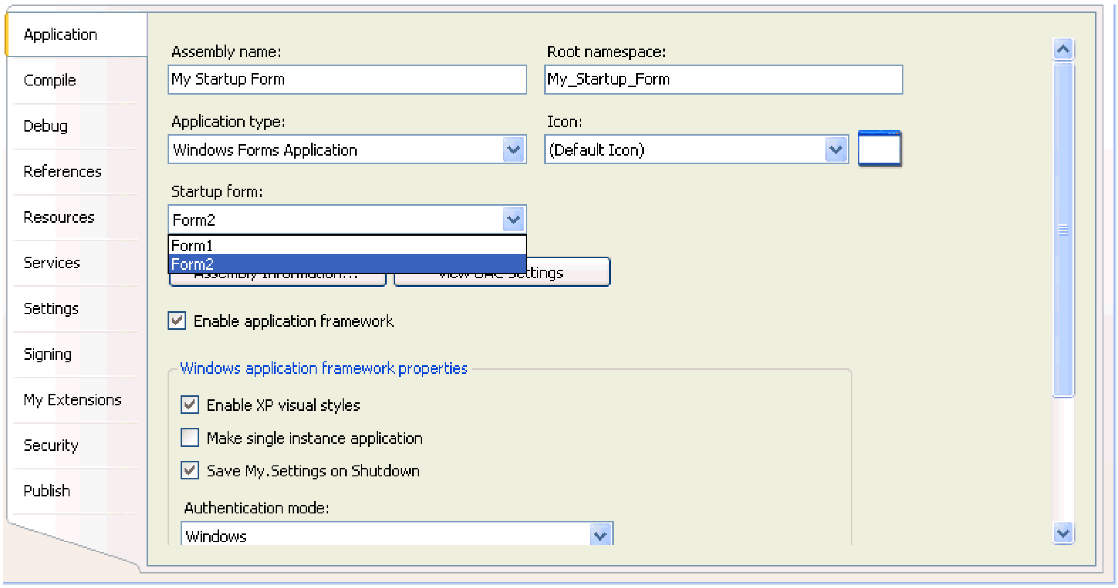
خطوة إضافية للأمام: تحديد الكائن الذي يظهر في بداية تنفيذ البرنامج

إذا احتوى التطبيق الذي صممناه على أكثر من نموذج (أكثر من فورم) يقوم برنامج الفيجوال بيسك بجعل النموذج الأول (Form1) هو النموذج الذي يظهر أولاً عند تنفيذ البرنامج، لكننا نستطيع تغيير ذلك من خصائص التطبيق ونجعل نموذج آخر يظهر بدلاً منه. كما في المثال التالي:

١- ننشئ مشروع جديد باسم My Startup Form

٢- نضيف نموذج ثاني للمشروع.

٣- لتغيير النموذج الذي يظهر في بداية تشغيل التطبيق نذهب إلى القائمة Project ونختار منها My Startup Properties لتظهر لنا هذه النافذة التي نختار منها نموذج بداية التشغيل:



لاحظ الخيار Startup Form الذي نستطيع منه تغيير النموذج الذي يظهر في بداية التشغيل، جرب تغيير نموذج بداية التشغيل إلى Form2 ثم قم بتنفيذ البرنامج و لاحظ ظهور النموذج الثاني مباشرة.

خلاصة الفصل الرابع عشر

من اجل أن	قم بالتالي

إضافة نموذج جديد للبرنامج	من قائمة Project نختار Add Windows Form ثم نختار Add، نستطيع كذلك إظهار نفس قائمة Project عند الضغط Right-Click على اسم المشروع في أعلى يمين بيئة التطوير في Solution Explorer
الانتقال من نموذج إلى آخر	نستخدم Show أو ShowDialog كالتالي: form2.ShowDialog() أو نستخدم المجال My.Forms كالتالي: My.Forms.form2.ShowDialog() ونستطيع استخدام الكائن Me لإخفاء النموذج كالتالي: Me.Visible = False أو لإظهار النموذج كالتالي: Me.ShowDialog() مع ملاحظة أن Me يجب استخدامها ضمن النموذج نفسه، أي ضمن الكود الموجود بداخل نفس النموذج، ولا يمكن تطبيقها على نموذج ثاني غير الذي نحن فيه.
إضافة نموذج آخر للمشروع بواسطة الكود	نقوم أولاً بتعريف النموذج ثم نقوم بإضافة الخصائص الهامة: Dim form2 As New Form form2.Text = "النموذج الجديد"
تحديد مكان النموذج على سطح المكتب وقت	نقوم بتحديد مكان ظهور النموذج على سطح المكتب بواسطة الخاصية StartPosition فنختار منها مثلاً CenterScreen ليظهر النموذج في

تنفيذ البرنامج	وسط شاشة الكمبيوتر، أو نختار CenterParent ليظهر النموذج في وسط النموذج الأب (إذا وُجد).
تحديد مكان النموذج على سطح المكتب باستخدام الكود	نقوم بتغيير الخاصية StartPosition إلى Manual ثم نقوم بتعريف شكل مستطيل ليقوم هذا المستطيل بحفظ مساحة النموذج ومكان تواجده على الشاشة، بعدها نستخدم الخاصية DesktopBounds لتقوم برسم النموذج في المكان المُحدد على شاشة الكمبيوتر كالتالي: form2.StartPosition = FormStartPosition.Manual Dim Form2Rect As New Rectangle _ (200, 100, 300, 250) form2.DesktopBounds = Form2Rect
تكبير وتصغير النموذج وقت تشغيل البرنامج	لابد من تغيير الخاصيتين MaximizeBox و MinimizeBox إلى True في مرحلة تصميم البرنامج، وفي مرحلة التصميم نقوم بضبط الخاصية WindowState على FormWindowState.Minimized أو FormWindowState.Maximized أو FormWindowState.Normal
إضافة كائن إلى النموذج وقت تنفيذ البرنامج	نقوم أولاً بتعريف الكائن، ثم تحديد خصائصه ثم نقوم بإضافته إلى مجموعة الكائنات على النموذج: Dim button1 as New Button button1.Text = "الزر الجديد" button1.Location = New Point(20, 25) form2.Controls.Add(button1)

المحافظة على مسافة معينة من أحد حواف النموذج لكائن معين	إذا أردنا المحافظة على مسافة معينة لكائن معين من حافة من حواف النموذج فنستخدم الخاصية Anchor وتحديد الحافة التي نريد، مع ملاحظة أننا نستخدم المعامل Or إذا أردنا المحافظة على مسافة معينة مع أكثر من حافة: `Button1.Anchor = AnchorStyles.Bottom Or _ AnchorStyles.Right`
لصق الكائن بإحدى حواف النموذج	نستخدم الخاصية Dock لتحديد الحافة التي نريد أن يلصق الكائن بها: `PictureBox1.Dock = DockStyle.Top`
تحديد النموذج الذي يظهر عند تشغيل البرنامج	من قائمة Project نختار My Startup Properties ومن الخاصية Startup Form نختار النموذج الذي نريد أن يفتح أولاً عند تشغيل التطبيق.
نستطيع كذلك إنشاء تطبيق لا يحتوى على نماذج ولا على أيه كائنات مرئية بواسطة إنشاء مشروع جديد ونختار Console Application ثم نقوم بكتابة كود التطبيق في وحدات برمجية Modules حيث لا توجد نماذج ثم نقوم بتنفيذ البرنامج والذي سيظهر مثل برنامج سطر الأوامر Line Command الذي كان موجوداً في أنظمة التشغيل القديمة.	

الفصل الخامس عشر: إضافة تأثيرات رسومية ومتحركة

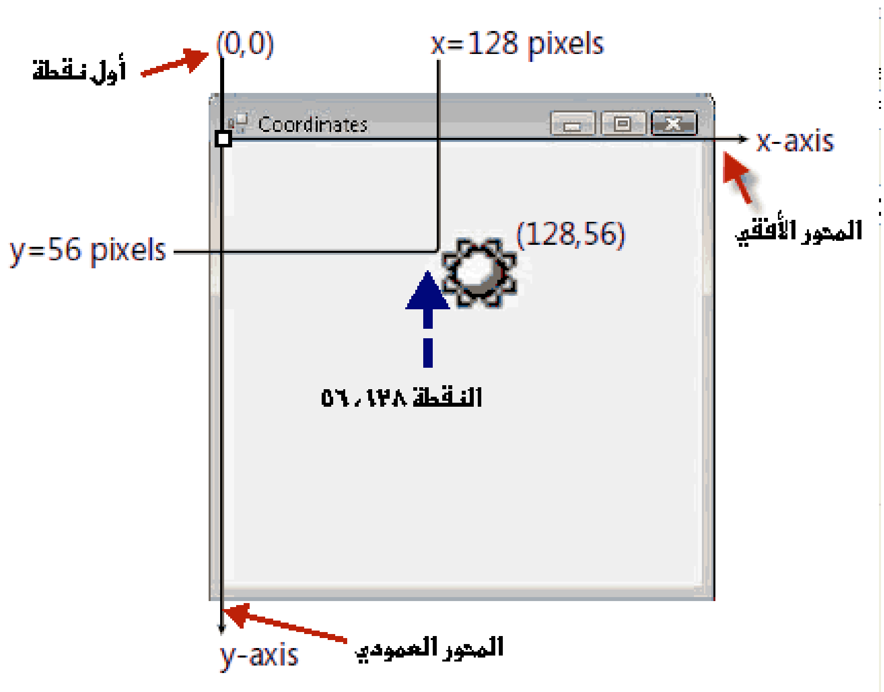
التعامل مع الرسوميات باستخدام مجال الأسماء System.Drawing

تعلمنا سابقاً كيف نقوم بإضافة الصور وغيرها من الرسوميات إلى النماذج بواسطة الفيجوال بيسك، فقمنا بالتعامل مع صناديق الصور وغيرها، الآن سنقوم بالتعامل مع ما يسمى بدوال الـــ GDI الموجودة ضمن مجال الأسماء System.Drawing التي تأتي ضمن إطارات العمل الدوت

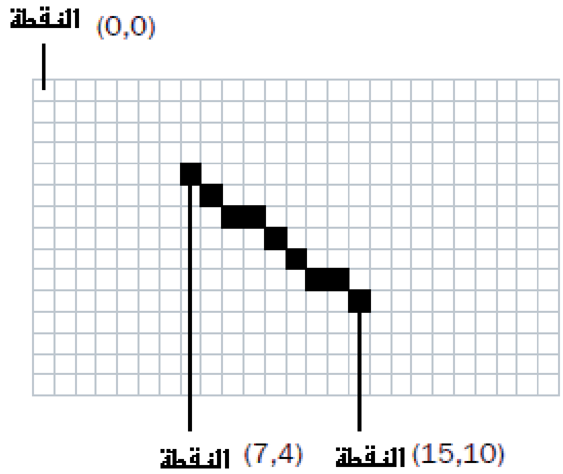
نت، والتي نستطيع من خلالها التعامل مع الصور وكذلك رسم الأشكال الثنائية الأبعاد بداخل نظام الويندوز.

فهم طبيعة النماذج

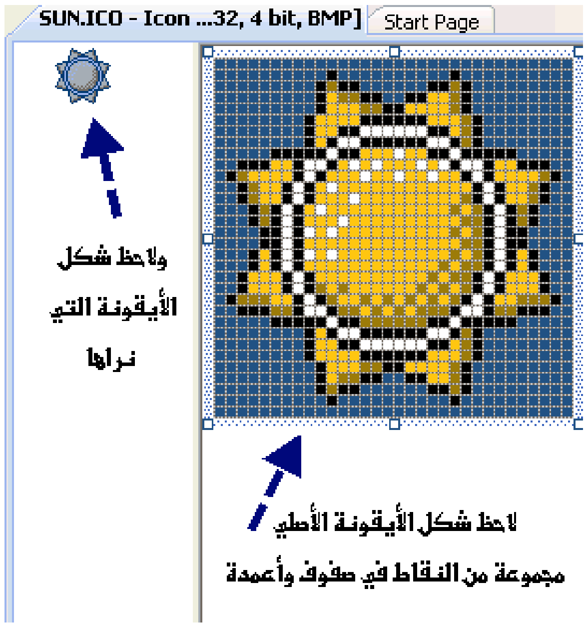
قبل أن نفكر في إنشاء الرسوميات على النموذج، يجب فهم تصميم النموذج أو ما يسمى From Layout، فكل نموذج له تصميمه الخاص حيث يبدأ هذا التصميم من نقطة معينه في أعلى يسار الشاشة، التخطيط أ والتصميم الطبيعي يكون على شكل صفوف وأعمدة تتكون من نقاط صغيرة جداً وهذا ما يسمى بالبكسل Pixel أي الطريقة النُقطية للرسم. هذه النقط تُرسم على المحورين الأفقي والعمودي يمثل المحور الأفقي الرمز x بينما يمثل المحور العمودي الرمز y، وعندما نحدد نقطة معينة على النموذج (الفورم) فهذه النقطة هي عبارة عن نقطة تلاقي المحور الأفقي مع المحور العمودي (x, y)، وعليه فأعلى نقطة في يسار النموذج تساوي القيمة (0, 0)، انظر الشكل التالي الذي قام برسمه المؤلف ليشرح لنا هذه الفكرة:



يقوم الفيجوال بيسك بالعمل بشكل آلي مع كرت الشاشة لتحديد كم عدد الصفوف والأعمدة وماذا يوجد في كل نقطة من نقاط الصفوف والأعمدة، لاحظ الشكل:



عند وجود نقطة واحدة فتعتبر نقطة فقط، لكن عند وجود مجموعة نقاط فهذا يعني أنه لدينا خط أو دائرة أو مستطيل أو غيرها من الأشكال الرسومية، سنتخيل الآن شاشة الكمبيوتر وهي تتكون من العديد من الصفوف الأفقية والأعمدة، ونتخيل كذلك كيف يقوم الكمبيوتر بخداعنا عندما يعرض لنا شكل من الأشكال (دائرة، مستطيل، خط مستقيم) حيث يعرض لنا مجموعة من النقاط المتناهية في الصغر ونرى هذه المجموعة من النقاط على أنها خط مستقيم أو دائرة لتوضيح الفكرة أكثر لاحظ الشكل التالي للأيقونة التي استخدمناها في التطبيق السابق:



الفئة System.Drawing.Graphics

يحتوي مجال الأسماء System.Drawing على العديد من الفئات التي تساعدنا على التعامل مع الرسوميات في برنامجنا، سنتعرف الآن إلى System.Drawing.Graphics التي تعنى برسم الأشكال على النماذج. يمكنك معرفة بقية الفئات بالرجوع إلى التعليمات المرفقة بالفيجوال بيسك ٢٠٠٨، الجدول التالي يحتوي على بعض الطرق Methods المتوفرة ضمن الفئة Graphics:

الشكل	الطريقة	الوصف
الخط المستقيم	DrawLine	لرسم خط يُوصل بين نقطتين
المستطيل	DrawRectangle	لرسم مستطيل أو مربع يُوصل بين أربع نقاط.
قوس (منحنى)	DrawArc	لرسم قوس أو خط منحنى، جزء من "دائرة"
دائرة/"قطع ناقص"	DrawEllipse	لرسم شكل دائري أو بيضاوي محدود بواسطة مستطيل.
مضلع	DrawPolygon	لرسم مضلع، شكل يحتوي على العديد من الأضلاع والزوايا تخزن قيمها بداخل مصفوفة.
منحنى	DrawCurve	هذا المنحنى نقوم بتحديد النقاط التي يمر فيها وخزنها في مصفوفة، بخلاف القوس في الطريقة DrawArc حيث يتم تحديد نقطتين فقط.

جميع الطرق أعلاه تقوم برسم أشكال فارغة على الفورم، وهناك طرق Methods أُخرى تقوم برسم أشكال مغطاة بالألوان، هذه الطرق تستخدم نفس الأوامر السابقة باستبدال Draw بالبادئة Fill فنكتب FillRectangle بدلاً من DrawRectangle. عند استخدام System.Drawing.Graphics لرسم الأشكال على النماذج لابد من استخدام (كائن وسيط) قلم Pen أو فرشاة Brush للرسم فعند رسم خط مستقيم أو شكل فارغ يمكننا استخدام القلم أما إذا أردنا رسم الأشكال المليئة بالألوان فلابد من استخدام الفرشاة. لنفرض الآن بأننا نريد أن نرسم خط مستقيم من النقطة ٢٠,٣٠ إلى النقطة ١٠٠,٨٠ على النموذج، لابد أولاً من تعريف كائن الرسومات Graphics object ثم نقوم بتعريف القلم (لأننا نريد رسم خط مستقيم)، ثم نحدد وظيفة كائن الرسومات نكتب الأمر بتنفيذ الرسم كالتالي:

```
Dim GraphicsFun As Graphics
Dim PenColor As New Pen(Color.Red)
GraphicsFun = Me.CreateGraphics
GraphicsFun.DrawLine(PenColor, 20, 30, 100, 80)
```

استخدام الحدث Paint التابع للنموذج للرسم

عند استخدامنا للكود أعلاه في رسم خط مستقيم على النموذج قام البرنامج برسم الخط لكن إذا قمنا بتصغير النموذج ثم فتحه مرة ثانية أو فتحنا صندوق حوار فوق الفورم سيختفي الخط المرسوم وإلى الأبد، ولحل هذه المشكلة لابد من استخدام الحدث Paint التابع للنماذج للرسم على النموذج هذا الحدث يقوم بالرسم على النموذج (الفورم)، وإذا تمت تغطية النموذج بصندوق حوار أو تم تصغير النموذج ثم فتحه لمرة ثانية نلاحظ وجود هذا الخط، لإن الحدث Paint يقوم بالرسم على النموذج كل مرة نفتح فيها النموذج (بعبارة أُخرى في حال استخدام الحدث Paint للرسم فلا تخف على الأشكال المرسومة على النموذج).

لنأخذ الآن مثالاً على رسم بعض الأشكال (خط مستقيم، مستطيل، شكل دائري) باستخدام الحدث Paint على النموذج:

١- ننشئ تطبيق جديد ونسميه My Draw Shapes

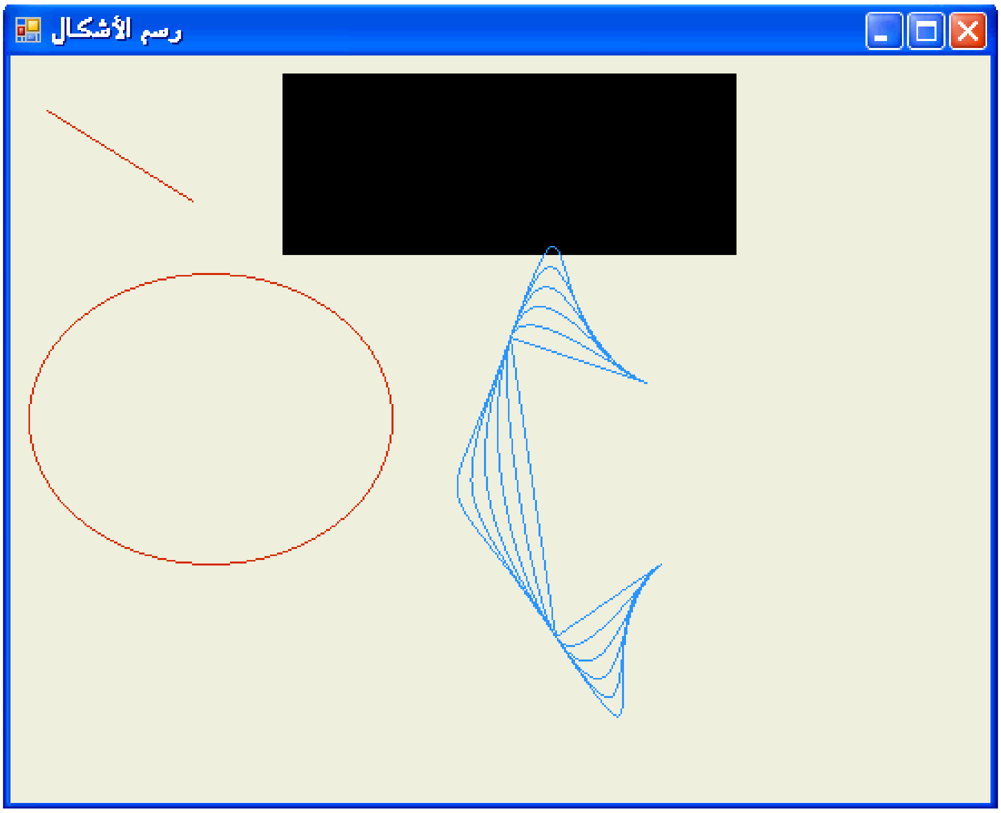
٢-نقوم بتكبير النموذج ضعف المساحة المعروضة، ونغير الخاصية Text للنموذج إلى "رسم الأشكال"

٣-ننتقل إلى منطقة الكود وفي أعلى يسار منطقة الكود (من قائمة General) نختار Form1 Events وننتقل لليمين ونختار الحدث Paint سيقوم الفيجوال بيسك بفتح الكود الخاص بالحدث Paint في منطقة الكود لتقم أنت بكتابة أوامِرُك الخاصة بهذا الحدث بين Private Sub و End Sub ، قم بكتابة هذا الكود:

```
'نقوم بتعريف كائن الرسومات
' بعد التعريف نقوم بتحديد مهمة الكائن وهي إنشاء الرسوم
Dim GraphicsFun As Graphics
GraphicsFun = Me.CreateGraphics
'نستخدم القلم الأحمر لرسم خط مستقيم وكذلك قوس جزء من دائرة
Dim PenColor As New Pen(Color.Red)
GraphicsFun.DrawLine(PenColor, 20, 30, 100, 80)
GraphicsFun.DrawEllipse(PenColor, 10, 120, 200, 160)
'نستخدم الفرشاة الخضراء لرسم مستطيل ملون
Dim BrushColor As New SolidBrush(Color.Green)
GraphicsFun.FillRectangle(BrushColor, 150, 10, 250, 100)
' نقوم برسم شكل كاردينالي قوسي باللون الأزرق وبأربع نقاط
'Create a blue cardinal spline curve with four points
Dim Points() As Point = {New Point(358, 280), _
New Point(300, 320), New Point(275, 155), New Point(350, 180)}
For tension As Single = 0 To 2.5 Step 0.5
```

```
    GraphicsFun.DrawCurve(Pens.DodgerBlue, Points, tension)
Next
```

في الكود أعلاه ومن أجل تنفيذ الرسومات في الفيجوال بيسك نقوم بتعريف كائن الرسومات كما في السطر الأول من الكود بعدها نحدد طبيعة عمل هذا الكائن، ثم استخدمنا القلم الأحمر لرسم خط مستقيم ودائرة، واستخدمنا الفرشاة لرسم مستطيل ملون، قمنا بعدها برسم شكل كارديناالي بخمسة خطوط وأربع نقاط باستخدام القلم الأزرق. لاحظ شكل النموذج بعد تنفيذ البرنامج:

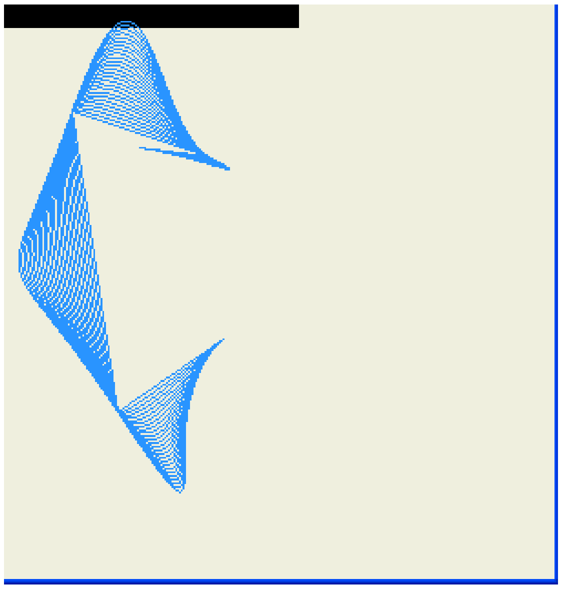


لتعديل الشكل الكارديناالي نتحكم في المتغيرات التابعة له وهي النقاط (وهي زوايا الشكل) وكذلك الخطوات (وهي عدد خطوط الشكل) فإذا عدلنا الكود التابع له كالتالي:

```
Dim Points() As Point = {New Point(358, 280), _
New Point(300, 320), New Point(275, 155), New Point(350, 180),
New Point(312, 175)}
For tension As Single = 0 To 2.5 Step 0.1
```

```
    GraphicsFun.DrawCurve(Pens.DodgerBlue, Points, tension)
Next
```

سيصبح الشكل كا في هذه الصورة:



ملاحظة: توجد نسخة هذا التطبيق في المرفق رقم ٠٤٤.

إضافة الرسم المتحركة Animations إلى برامجك

إضافة الأشكال والصور النقطية إلى برامجك تعتبر ميزة جميلة للفيجوال بيسك، ولكن كلما تحركت هذه الرسوم كلما كان أفضل، نستطيع تحريك الرسوم على النموذج وكذلك تغيير أحجامها ومساحتها أثناء التحريك.

تحريك الكائنات على النموذج:

في فيجوال بيسك ٦ كنا نستخدم الأمر Move لتحريك الكائنات على النموذج، هذا الأمر غير مدعوم في فيجوال بيسك ٢٠٠٨، نستطيع تحريك الكائنات على النماذج في نسخة ٢٠٠٨ باستخدام الطرق Methods والخصائص التالية:

الطريقة	الوصف

Left	نستخدم هذه الخاصية لتحريك الكائن أفقياً لليمين أو لليسار.
Top	تستخدم هذه الخاصية لتحريك الكائن عمودياً للأعلى أو للأسفل.
Location	تستخدم هذه الخاصية لتحريك الكائن إلى مكان محدد.
SetBounds	هذه الخاصية تحدد مساحة ومكان جديد للكائن.

فمثلاً نستخدم Lift لتحريك الكائن كالتالي:

```
object.Left = horizontal
```

فـــ object هو الكائن الذي تريد تحريكه ونستبدل horizontal بالعمود العمودي التي نريد أن يكون الكائن عنده (يجب العِلم بأن البرنامج يقسم النموذج إلى مجموعة من الخطوط الأفقية والعمودية المتناهية في الصغر) التحريك كالتالي:

```
PictureBox1.Left = 300
```

فسينتقل صندوق الصورة إلى العمود رقم ٣٠٠، ولكن ماذا إذا أردنا تحريك الكائن بمقدار محدد (مثلاً تحريك الكائن إلى بعد خمسين نقطة بغض النظر عن معرفة رقم العمود الأفقي الذي نريد أن نصل إليه) عندها نستخدم الأمر التالي:

```
PictureBox1.Left = PictureBox1.Left + 50
```

بنفس الطريقة يمكننا تحريك الكائن للأعلى وللأسفل كالتالي:

```
PictureBox1.Top = 150
```

أو تحريكه لمسافة معينة:

```
PictureBox1.Top = PictureBox1.Top + 30
```

الخاصية Location

لتحريك الكائن أفقياً وعمودياً نستطيع استخدام مزيج من الأمرين السابقين Left و Top فنقوم بتحديد المكان الجديد للصورة بوضع النقاط الجديدة لأعلى يسار الصورة في الكود على النحو التالي:

```
PictureBox1.Left = 300
PictureBox1.Top = 200
```

فسيقوم البرنامج بتحريكها مباشرة إلى المكان المناسب، لكن في فيجوال بيسك ٢٠٠٨ لا يُفضل استخدام مثل هذه الطريقة خاصة إذا كان لابد من تحريك الكائن مئات أو ألاف المرات على النموذج، في مثل هذه الحالة لابد من استخدام الخاصية Location وتحديد النقطة العمودية والأفقية للمكان الذي نريد نقل الكائن إليه كالتالي:

```
object.Location = New Point(horizontal, vertical)
```

فـــ object هو الكائن الذي نريد تحريكه، و horizontal النقطة الأفقية، و vertical النقطة العمودية، مثال:

```
PictureBox1.Location = New Point(300, 200)
```

ماذا إذا أردنا تحريك الكائن لمسافة معينة بدون تحديد النقاط الجديدة له، نستخدم هذا الكود:

```
PictureBox1.Location = New Point(PictureBox1.Location.X - 50, _
PictureBox1.Location.Y - 40)
```

الكود أعلاه يقوم ٥٠ نقطة لليسار وكذلك ٤٠ نقطة للأعلى.

إنشاء الرسوم المتحركة Animations باستخدام المؤقت Timer

في حالة استخدام المؤقت لتنظيم عملية تحريك الكائن على النموذج فهذا يعني ميزة جديدة للتعامل مع الرسوم، ومع كل وقت محدد سيتحرك الكائن إلى نقطة جديدة (هل تتذكر الفصل السابع الذي تعاملنا فيه مع المؤقت Timer) عند استخدام الكائن Timer لتحريك الرسوم يجب تعديل الخاصية Interval التابعة له إلى سرعة مناسبة فمثلاً خُمس الثانية يساوي ٢٠٠ وعُشر الثانية يساوي ١٠٠ لإن البرنامج يقسم الثانية إلى ١٠٠٠ Interval، نستطيع دمج كود تحريك الكائن

وكود المؤقت مع كود شرطي لتحديد نهاية النموذج باستخدام الخاصية Size التابعة للنموذج واستخدام الخاصيتين Top و Left اللتان تحدثنا عنهما سابقاً. بمزج الأكواد المذكورة آنفاً نستطيع تحريك الكائن وبسرعة مناسبة حتى يصل إلى نهاية الفورم ثم يتوقف الكائن عن الحركة. في المثال التالي سنتعلم كيف نقوم بتحريك كائن مربع الصورة الذي يحتوي على أيقونة الشمس Sun.ico التي استخدمناها سابقاً باستخدام الخاصية location ومؤقت وسنتعلم كيفية استخدام الخاصية Top لتحديد أعلى النموذج والخاصية Size.Height لتحديد أسفل النموذج:

١-نقوم بإنشاء تطبيق جديد باسم My Moving Icon

٢-نضيف اثنين من الأزرار إلى أسفل يسار النموذج ونضيف مربع صورة إلى أسفل يمين النموذج، ونضيف مؤقت Timer.

٣-نقوم بتعديل الخاصية Text للزر الأول إلى "تحريك للأعلى"، وللزر الثاني إلى "تحريك للأسفل"، وللنموذج إلى "التحريك البسيط للرسوم" ونغير الخاصية Interval للمؤقت إلى ٧٥ ونضيف الصورة Sun.ico إلى مربع الصورة، ونعدل الخاصية SizeMode لمربع الصورة إلى StretchImage. سيكون النموذج كما في الشكل التالي:



نقوم بالضغط نقرتين على الزر "تحريك للأعلى" ونكتب هذا الكود في منطقة الكود:

```
GoingUp = True
```

```
Timer1.Enabled = True
```

الكود يفعل الأمر GoingUp ويفعل المؤقت، الأمر GoingUp موجود كاملاً في الحدث Tick التابع للمؤقت، ستلاحظ خط أزرق متعرج تحت الأمر GoingUp لأننا لم نقم بتعريفه بعد. في أعلى منطقة الكود وتحت Public Class Form1 نكتب الكود التالي لتعريف الأمر GoingUp:

```
Dim GoingUp As Boolean
```

نقوم بالضغط نقرتين على الزر "تحريك للأسفل" ونكتب الكود التالي:

```
GoingUp = False
Timer1.Enabled = True
```

وكذلك ننقر نقرتين على المؤقت ونكتب الكود التالي:

```
If GoingUp = True Then
   'تحريك الصورة للأعلى
   If PictureBox1.Top > 10 Then
PictureBox1.Location = New Point _
(PictureBox1.Location.X - 10, _
PictureBox1.Location.Y - 10)
   End If
Else
   'تحريك الصورة للأسفل
   If PictureBox1.Top < (Me.Size.Height - 75) Then
PictureBox1.Location = New Point _
(PictureBox1.Location.X + 10, _
PictureBox1.Location.Y + 10)
```

```
    End If
End If
```

الكود أعلاه يقوم بالتأكد من أن المتغير GoingUp مضبوط على True فإذا كان كذلك يقوم بتحريك مربع الصورة بمقدار عشر نقاط لليسار والأعلى وذلك كل ٠,٧٥ ثانية وهو الوقت المحدد في خاصية Interval للمؤقت، يقوم البرنامج بتحريك مربع الصورة للأعلى ولليسار حتى إذا لم يبقى في النموذج إلا أقل من عشر نقاط من الأعلى وعشر نقاط من اليسار يتوقف البرنامج عن التحريك. وإذا كان المتغير GoingUp مضبوط على False (ويحدث هذا عند الضغط على الزر "تحريك للأسفل") يقوم البرنامج بتحريك الصورة إلى أسفل يمين الصورة بمقدار عشر نقاط لليمين وعشر نقاط للأسفل كل ٠,٧٥ ثانية. وقمنا بطرح الرقم ٧٥ من إجمالي طول النموذج حيث يقصد بالضمير Me بالنموذج Form1 لتبقى الصورة ظاهرة ولا تختفي.

قم بتنفيذ البرنامج لترى كيف يقوم البرنامج بتحريك الصورة، وإذا أردنا التحريك بسرعة أكبر نقوم بتعديل الخاصية Interval التابعة للمؤقت إلى رقم أصغر.

ملاحظة: توجد نسخة من التطبيق أعلاه في المرفق ٠٤٥.

تصغير وتكبير كائن ما خلال مرحلة تنفيذ البرنامج

بالإضافة إلى الخاصيتين Top و Lift لتحريك الكائنات على النموذج توجد الخاصيتين Height و Width لتكبير وتصغير الكائنات على النموذج، لنأخذ مثالاً لشرح الفكرة:

١- ننشئ تطبيق جديد بالفيجوال بيسك ٢٠٠٨ ونسميه My Zoom In

٢- نضيف صندوق صورة للنموذج ونضعه في أعلى يسار النموذج، سنضيف صورة إلى صندوق الصورة على أساس أن هذه الصورة اُلتقطت في الفضاء ولأن الصورة التقطت في الفضاء سنجعل خلفية النموذج BackColor باللون الأسود، وسنضع صورة النجوم خلفية للنموذج BackgroundImage (عليك أن تفرق بين لون خلفية النموذج BackColor وبين صورة كخلفية للنموذج BackgroundImage)، وعلية فالخاصية BackColor التابعة للنموذج سنضبطها على اللون الأسود Black أما خاصية الصورة الخلفية فسنختار

الصورة المرفقة بالمرفق ٠٤٦ باسم space.jpg، أما الصورة التي ستكون في صندوق الصورة فهي صورة للأرض Earth.jpg مرفقة كذلك، ونضبط الخاصية SizeMode على StretchImage.

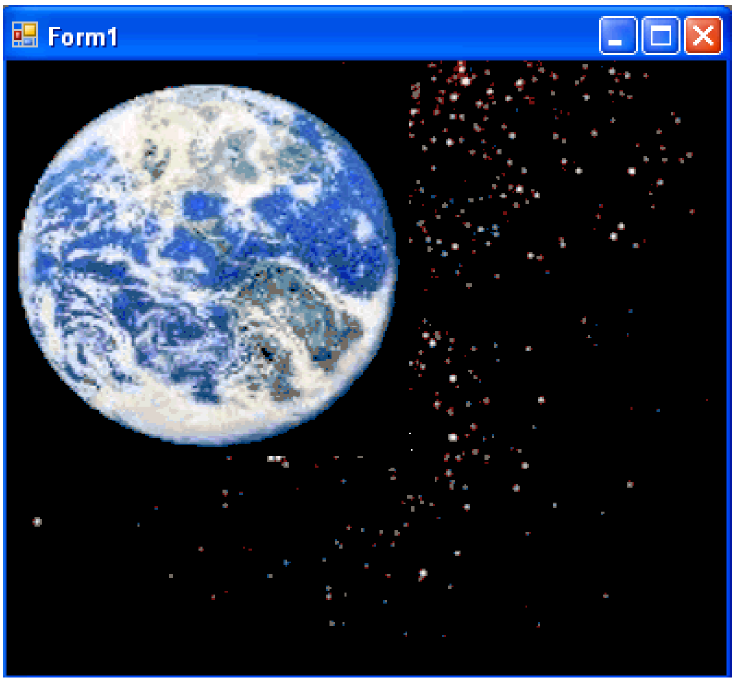
٣-نقوم بالضغط نقرتين على صندوق الصورة وكتابة الكود التالي في الحدث Click:

```
PictureBox1.Height = PictureBox1.Height + 15
PictureBox1.Width = PictureBox1.Width + 15
```

الكود أعلاه يقوم بزيادة ارتفاع وعرض صندوق الصورة بمقدار ١٥ نقطة ولأن مساحة الصورة بداخل صندوق الصورة تكبر بشكل مطاطي بسبب الخاصية SizeMode فستكبر الصورة مع صندوق الصورة في نفس الوقت بمقدار ١٥ نقطة، وكلما استرينا في الضغط على الصورة فإنها ستكبر أكثر فأكثر.

قم الآن بتنفيذ البرنامج وقم بالضغط على صورة الكرة الأرضية، ماذا تلاحظ.

بعد ما يقرب من ١٠ نقرات Clicks على الصورة ستكون بهذا الحجم:



ملاحظة: نسخة من المشروع أعلاه في المرفق رقم ٠٤٦.

خطوة إضافية للأمام: تعديل شفافية النموذج

باستخدام دوال الـــ GDI تستطيع تحقيق العديد من المهام التي كانت من المستحيلات في النسخ الأقدم من فيجوال بيسك (لعل متعصبي فيجوال بيسك ٦ ينكرون ذلك)، فيمكننا مثلاً تعديل شفافية النموذج (الفورم) حتى تكون شفافة جداً نستطيع أن نرى من خلالها ما تحتها (مثل هذه الشاشات الشفافة نراها مع برامج محاربة الفيروسات، أو برامج التنزيل من الإنترنت أو برامج تعديل الصور وغيرها من البرامج)، من ضمن خواص النموذج (الفورم) الخاصية Opacity وهي الخاصية التي تعنى بشفافية النموذج وقيمتها تتراوح بين صفر إلى ١٠٠، فصفر تعني شفاف جداً (غير مرئي) و ١٠٠ تعني غير شفاف، لنأخذ مثالاً على ذلك:

١- ننشئ تطبيق بالفيجوال بيسك ٢٠٠٨ ونسميه My Transparent Form

٢- بعد فتح النموذج نضيف اثنين أزرار له، ونعدل الخاصية Text لأحدهما على "شفافية أكثر" والآخر على "استعادة"

٣- نضغط نقرتين على الزر الأول ونكتب الكود:

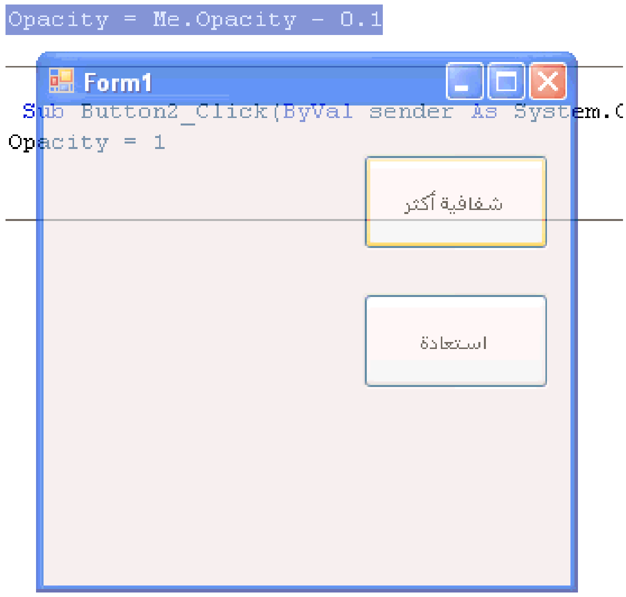
```
Me.Opacity = Me.Opacity - 0.1
```

في الكود أعلاه قمنا بتحديد الشفافية للنموذج على أساس الشفافية الموجودة حالياً مطروحاً منها عشرة بالمائة (لأن الشفافية يُعبر عنها بالنسب المئوية بين ٠ و ١٠٠) 0.1، نستطيع كتابة شفافية معينة للنموذج بأن نكتب الكود Me.Opacity = 0.75 لتكون الشفافية ٧٥% لكننا كتبنا الكود كما في أعلاه حتى إذا ضغطنا على الزر مرة أُخرى تنقص الشفافية أكثر بمقدار عشرة بالمائة حتى نصل إلى الضغطة العاشرة والتي يكون فيها النموذج غير مرئي.

٤- ننقر نقرتين على الزر الثاني ونكتب الكود:

```
Me.Opacity = 1
```

بعد تنفيذ البرنامج والضغط أربع مرات على الزر "شفافية أكثر" نشاهد هذا النموذج الشفاف:



ملاحظة: نسخة من التطبيق في المرفق ٠٤٧.

خلاصة الفصل الخامس عشر:

من اجل أن	قم بالتالي
رسم خطوط أو أشكال على النموذج	نستخدم إحدى الطرق الموجودة ضمن مجال الأسماء System.Drawing.Graphics كمثال الكود التالي يقوم برسم دائرة على النموذج: Dim GraphicsFun As Graphics GraphicsFun = Me.CreateGraphics Dim PenColor As New Pen(System.Drawing.Color.Red) GraphicsFun.DrawEllipse(PenColor, 10, _ 120, 200, 160)
رسم خطوط أو أشكال	نضع أكواد الرسم في الحدث Paint التابع للنموذج

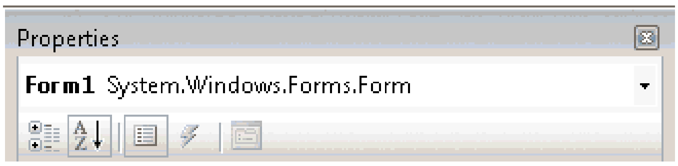
تستمر على النموذج ولا تختفي بمجرد تصغير النموذج	
تحريك الكائنات على النموذج	نقوم بتغيير مكان الكائن على النموذج باستخدام الخاصية Location الخاصة بالكائن من خلال الكود كالتالي: PictureBox1.Location = New Point(300, 200)
تحريك كائن بشكل مستمر	نستخدم مؤقت لتحريك الكائن على النموذج، ثم نستخدم الخاصيتين Top و Lift أو الخاصية Location لتحريك الكائن ولزيادة سرعة التحريك نتحكم بالخاصية Interval التابعة للمؤقت Timer.
تكبير أو تصغير كائن خلال مرحلة تشغيل البرنامج	نستطيع ذلك بتعديل الخاصية Height و Width من خلال الكود.
تحديد لون كخلفية على النموذج (الفورم)	نستطيع ذلك من الخاصية BackColor التابعة للنموذج.
استخدام صورة كخاصية على النموذج	نستطيع ذلك من خلال الخاصية BackgroundImage التابعة للنموذج.
تعديل شفافية النموذج	نستطيع ذلك من خلال الخاصية Opacity التابعة للنموذج، وذلك بتغيير القيمة بين ٠ و ١٠٠.

الفصل السادس عشر: الوراثة (وراثة النماذج وتصميم الفئات)

من أهم الميزات التي تهم المبرمجين والمطورين هي البرمجة كائنية التوجه، هذه الطريقة من عبارة عن تسهيل وتطوير للبرمجة بمفهومها التقليدي. فدعم البرمجة الكائنية التوجه بدأ من النسخة ٦ للفيجوال بيسك لكن كان هناك بعض القصور في هذا الدعم مثل عدم دعم الوراثة، وهي عملية نقل خصائص ومظهر نموذج معين أو فئة معينة إلى فئات أُخرى، بدأ دعم الوراثة من نسخة فيجوال بيسك ٢٠٠٢ بمعنى أننا نستطيع بناء وتصميم نموذج معين لمرة واحدة ثم نقل مظهره وخصائصه إلى نموذج آخر، ونستطيع تصميم فئات معينة ثم وراثة خصائصها وأحداثها إلى فئة جديدة، تم تطوير دعم الوراثة في الفيجوال بيسك ٢٠٠٨ بشكل أكبر.

وراثة نموذج باستخدام الفيجوال بيسك ٢٠٠٨

عند إضافة نموذج (فورم) لتطبيقنا يقوم الفيجوال بيسك ٢٠٠٨ بعرضه في نافذة الخصائص بهذا الشكل:



قم بقراءة ما في الصورة أعلاه، كرر ذلك، ماذا ستلاحظ، ستلاحظ بأننا نقوم بوراثة شكل وخصائص النموذج من System.Windows.Forms.Form، بجانب هذا يسمح لنا الفيجوال بيسك بتعديل نموذج معين ثم وراثة خصائصه مرة أُخرى، لنأخذ مثال على ذلك:

١- نقوم بإنشاء تطبيق جديد باسم My Form Inheritance.

٢- قم بإضافة اثنين أزرار إلى النموذج وقم بتغيير خاصية Text لأحدهما إلى "نعم" والآخر إلى "لا"، وكذلك قم بتغيير خاصية Text للنموذج إلى "نافذة حوار".

٣- قم بالنقر نقرتين على الزر "نعم" وقم بكتابه الكود:

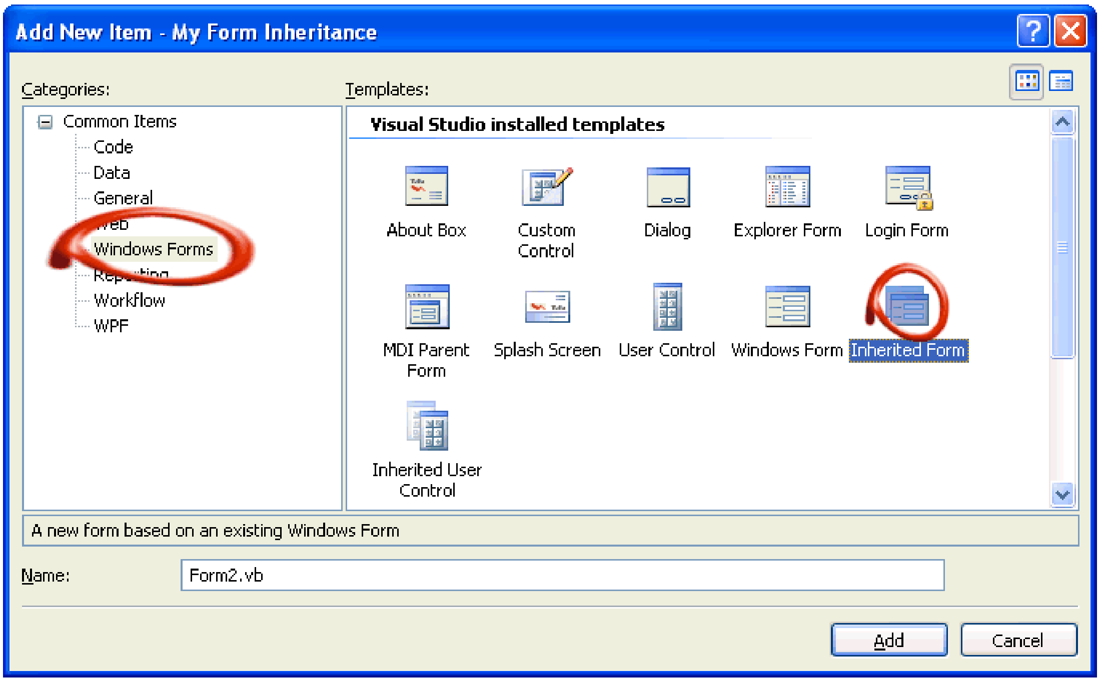
```
MsgBox("لقد قمت بالنقر على نعم")
```

٤- وقم كذلك بالنقر نقرتين على الزر "لا" وإضافة الكود التالي:

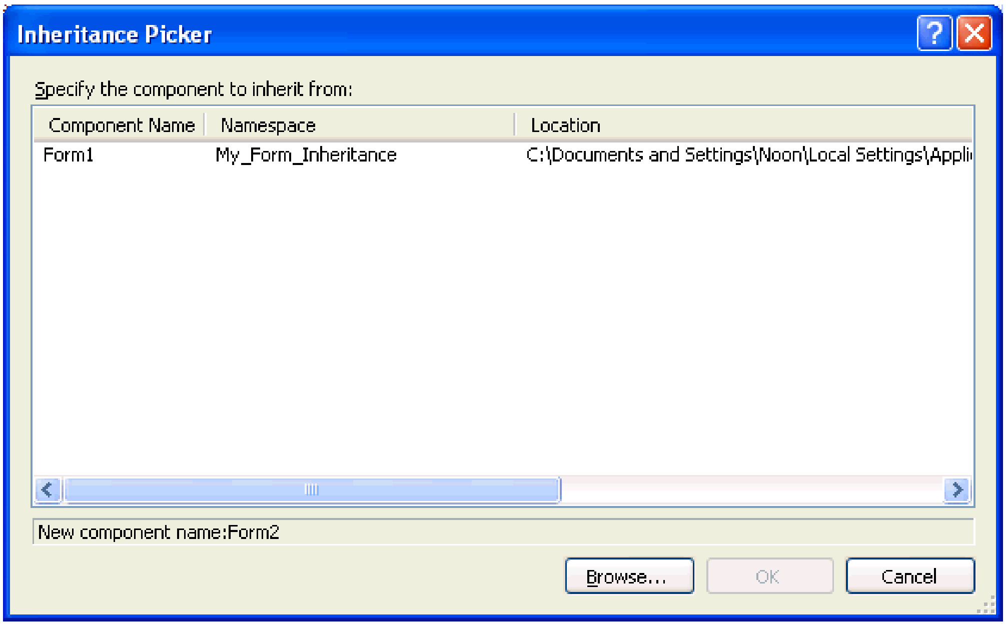
MsgBox("لقد قمت بالنقر على لا")

سنقوم الآن بوراثة هذا النموذج من نموذج آخر كالتالي:

٥- من Solution Explorer نضغط Right-Click على اسم المشروع ثم نختار Add ثم New Item تظهر الشاشة التالية:

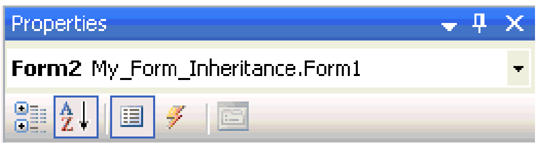


نختار منها Inherited From كما هو موضح بالشكل، ستظهر لنا شاشة اختيار النموذج الذي نريد الوراثة منه كالتالي:



وبما أن النموذج الذي نريد وراثة خصائصه موجود في نفس المشروع فنختاره من القائمة الظاهرة أعلاه وهو "Form1" أما إذا وراثة نموذج ليس من ضمن ملفات المشروع فيمكننا الضغط على الزر Browse ثم البحث عن ذلك النموذج واختياره لوراثته، لكن لابد أن يكون هذا النموذج على هيئة Dll وهي صيغة يُمكننا حفظ تطبيقاتنا بها إذا أردنا وراثة الخصائص لاحقاً.

على العموم نختار الآن Form1 ثم نضغط Ok، سيقوم الفيجوال بيسك بإضافة نموذج جديد للتطبيق وسيكون نفس النموذج السابق الذي قمنا بوراثته، لا يوجد فرق بين النموذجين ولكن هناك أيقونات صغيرة (أقفال) على الأزرار تبين لنا بأننا لا نستطيع تغيير هذه الأزرار على النموذج، إذا لاحظنا نافذة الخصائص سنرى:



لا حظ بأن نافذة الخصائص تبين لنا بأن هذا النموذج يقوم بوراثة خصائصه من النموذج Form1، ويجب العلم بأنه وبالرغم من أننا لا يمكننا التعديل على النموذج الإبن يمكننا إضافة مكونات جديدة له (لا يمكن تعديل المكونات السابقة) لكن يمكننا إضافة مكونات جديدة فيمكننا إضافة أزرار جديدة ومكونات أُخرى جديدة للنموذج، ولنجرب إضافة زر جديد ثم كتابة الكود التالي في الحدث Click التابع للزر الجديد:

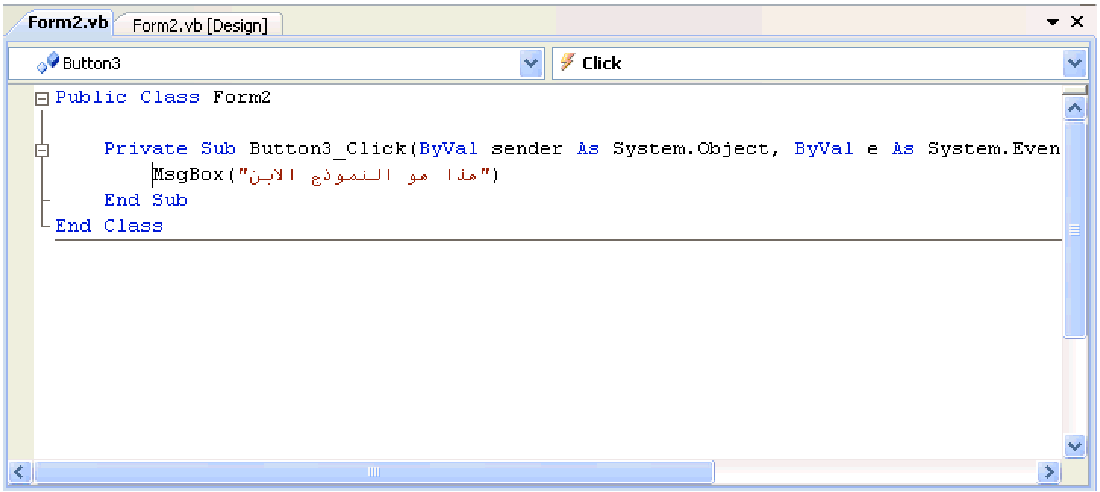
MsgBox("هذا هو النموذج الابن")

نقوم الآن بالتعديل على خصائص المشروع بالنقر Right-Click على المشروع ثم اختيار Properties ومن النافذة التي تظهر نختار Startup Form ومنها نختار Form2 ثم نقوم بتنفيذ البرنامج ليظهر النموذج "الإبن" أولاً، جرب انقر على الأزرار "نعم" و "لا"، جرب انقر على الزر الثالث.

ملاحظة: إذا أردنا وراثة نموذج معين في مشروع جديد فلابد أن تكون صيغة النموذج هي Dll (أما إذا كانت الوراثة في نفس التطبيق فلا ضير أن نرث النموذج بشكل مباشر بدون تحويله إلى صيغة Dll) ولذلك نجرب تصميم نموذج جديد ولتحويله إلى هيئة Dll من خصائص التطبيق نختار Application ومنه نختار Application Type ومنه نختار Class Library وبهذه الحالة سيقوم الفيجوال بيسك بتحويل النموذج إلى ملف Dll نجده تحت المجلد bin\Debug هذا الملف نستطيع الاحتفاظ به لوراثته لاحقاً في مشاريعنا الأُخرى.

تصميم فئة جديدة Class

عند وراثة نموذج معين فهذا يعني وراثة فئة معية لان كل نموذج هو عبارة عن فئة، عندما قمنا بالوراثة في المثال السابق وفتحنا نافذة الكود للنموذج الثاني يظهر لنا هذا الكود:



فأول سطر في الكود يوضح لنا بأن النموذج الثاني Form2 هو عبارة عن فئة Class، عندما يقوم الفيجوال بيسك بوراثة نموذج معين هو يقوم بوراثة الفئة Class حيث يقوم بالربط بين الفئة

هذا كل ما في الأمر، هذا يعني لنا بأننا نستطيع تصميم فئات جديدة Form1 والفئة Form2 ووراثة هذه الفئات، لاحظ الكود التابع للزر الثالث في الصورة أعلاه، فالزر الثالث لا يرث خصائصه من الفئة Form1، لفهم طبيعة العلاقة من جديد، فالنموذج الثاني يرث خصائصه من النموذج الأول والنموذج الأول يرث خصائصه من الفئة System.Windows.Forms.Form والفئة System.Windows.Forms ترث خصائصها من إطارات عمل الدوت نت DotNet Framework. (المشروع بالمرفق رقم ٠٤٨).

ملاحظة: بالإضافة إلى الوراثة بواسطة اختيار خصائص المشروع ثم New Item يمكننا وراثة فئة معينة بواسطة الكود، باستخدام الأمر Inherits ويجب أن نضع الأمر في أعلى منطقة الكود في أول سطر من الكود، استخدام الكود في الوراثة سيسهل لنا العديد من المهام، قبل نهاية الكتاب سنجد أمثلة تستخدم هذه الطريقة.

الآن قد تتساءل كيف نقوم بتصميم الفئات، وكيف نقوم بتطوير الفئات ثم بوراثتها وكيف نقوم بوراثة فئة من فئة أُخرى ثم تطويرها ووراثتها مرة أُخرى، في هذا الفصل سنعمل سوياً على تصميم فئة Class.

ملاحظة: موضوع الوراثة وكذلك البرمجة كائنية التوجه موضوع ليس مهم ومفيد ومعقد بعض الشيء وليس بالأمر السهل، تم تسهيل عملية الوراثة في فيجوال بيسك ٢٠٠٨، فالوراثة تسهل علينا عملية كتابة الأكواد فنقوم بكتابتها مرة واحدة ثم وراثتها، فالبرمجة كائنية التوجه تفيدنا في معرفة الكثير من الميزات التي يتميز بها الفيجوال بيسك ٢٠٠٨ مثل LINQ وغيرها من التقنيات الجديدة.

إضافة فئة Class جديدة إلى مشروعك

الفئة Class هي عبارة عن وعاء حاضن لكائن برمجي أو أكثر، يقوم محرر الكود بتلوينها باللون الأزرق، فيجوال بيسك بعد تعريف الفئة وإضافة الكائن البرمجي إليها سيتضمن هذا الكائن خصائص وأحداث وطرق Methods، مثل الكائنات التي نضيفها للنموذج، لإضافة فئة جديدة لبرنامجنا من مستكشف المشروع نختار Add ثم New Item ثم نختار Class ثم نقوم بتعريف الفئة باستخدام الكود، سنأخذ الآن مثال تطبيقي لإنشاء وتعريف فئة تحت اسم Person تقوم هذه

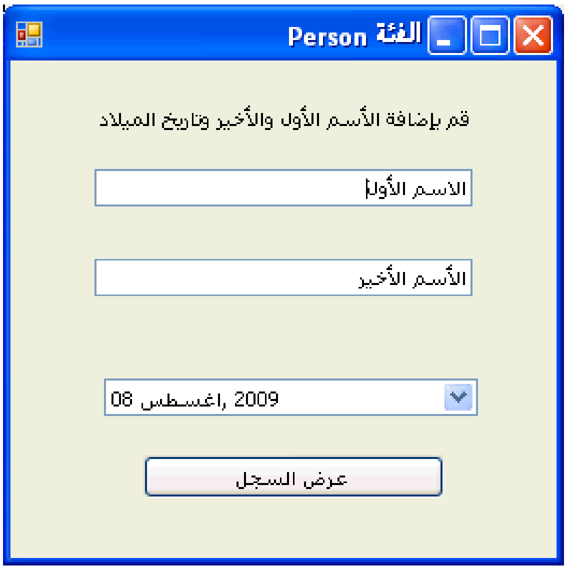
الفئة بأخذ الاسم الأول والأخير مع تاريخ الميلاد للشخص، وتقوم بحفظ البيانات في خصائص الفئة، سوف نقوم بإضافة طُرق Methods لحساب عمر الشخص بمجرد معرفة تاريخ ميلاده، سنتعلم عبر هذا التطبيق كيف نصمم فئاتُنا الخاصة وكذلك كيف نستخدم ونستفيد من الإجراءات المنبثقة من الفئات في مرحلة الكود.

مثال الفئة Person

١- ننشئ مشروع جديد باسم My Person Class

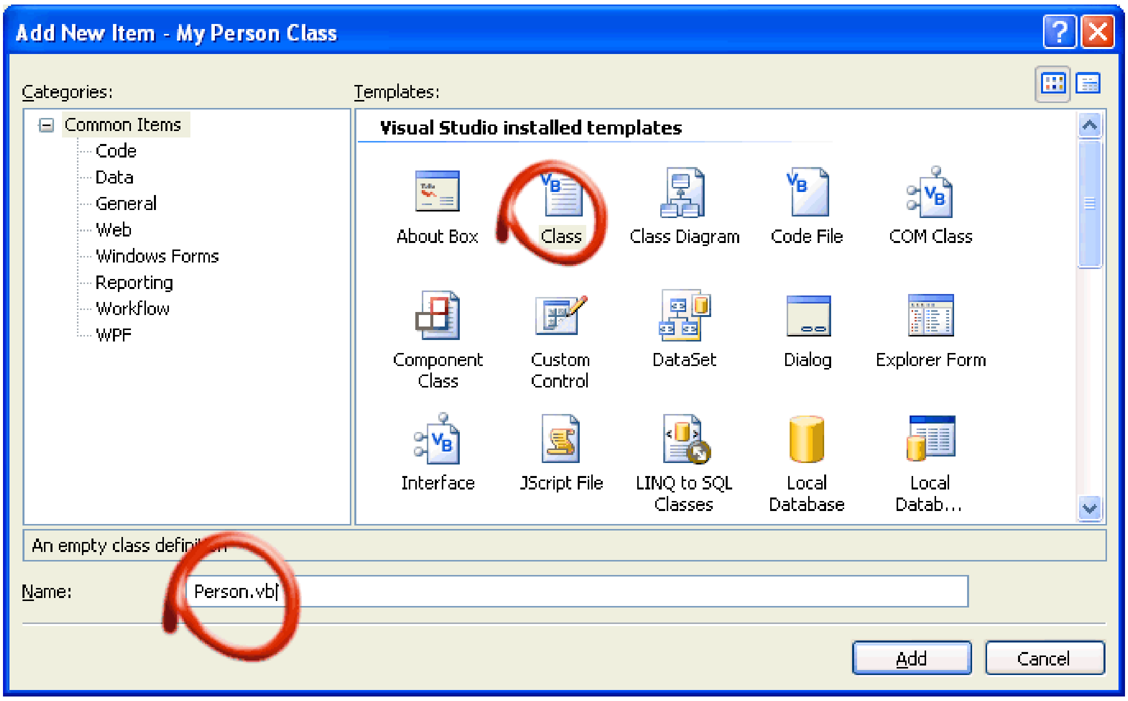
٢- نقوم بإضافة ليبل ثم اثنين صناديق نص ثم أداة الوقت والتاريخ Date and Time Picker ثم زر Button

٣- نقوم بتعديل الخصائص ليظهر النموذج والأدوات كما في الشكل التالي:



هذا النموذج غير مرتبط بقواعد البيانات، لذلك فهو يقوم بعرض سجل واحد فقط، سوف نتعلم في الفصل الثامن عشر عن الربط مع قواعد البيانات.

٤- نقوم بإضافة فئة Class للمشروع، نختار Add ثم New Item ثم نختار من النافذة التي تظهر الفئة Class ونسميها Person.vb ثم نضغط Add.



سيقوم الفيجوال بيسك بإضافة الفئة إلى ملفات المشروع ثم سيقوم بفتح صفحة الكود الخاصة بالفئة، وسنقوم الآن بكتابة الأكواد الخاصة بالفئة Person بحسب الترتيب التالي: تعريف متغيرات الفئة ثم كتابة الخصائص ثم الطرق Methods ثم إنشاء الكائنات التي تعتمد على هذه الفئة.

دعونا لنبدأ بكتابة كود الفئة خطوة بخطوة:

١- تعريف متغيرات الفئة

لابد أن نقوم بكتابة جميع الأكواد بين Public Class Person و End Class ولتعريف المتغيرات لابد أن نقوم بتعريفهم بعد Public Class Person مباشرة، فنكتب بعدها في السطر الثاني مباشرة التعريفات:

```
Private Name1 As String

Private Name2 As String
```

قمنا بتعريف متغيرين على هيئة String (نصوص)، لقد قمنا بتعريف المتغيرين بالكلمة Private لكي نستخدمهم بداخل الفئة Person فقط وغير قابلين للاستعمال على طول المشروع وعرضه، وهذه التقنية يقوم بها معظم مبرمجي الفيجوال بيسك حيث يقومون بتعريف المتغيرات

بداخل الفئات على أنها متغيرات خاصة وليست عامة، ولنأخذ مثال على الفئة Form التي قام مبرمجي مايكروسوفت بتعريف متغيراتها بنفس الطريقة ضمن إطارات الدوت نت.

٢- إنشاء الخصائص:

بعد تعريف المتغيرات لابد من إنشاء الخصائص، نكتب هذا الكود بعد تعريف المتغيرات:

```
Public Property FirstName() As String
```

بمجرد أن نضغط Enter بعد كتابة الجملة أعلاه يقوم الفيجوال بيسك بكتابة كود تلقائي من أجل تعبئته، فنحن قمنا بكتابة خاصية فقام الفيجوال بيسك بكتابة بقية كود الخاصية الذي لابد علينا من تعبئته لنقوم بتعريف الخاصية بشكل صحيح، بقية الكود يحتوي على بند Get والتي تعني ماذا سوف يرى المبرمج عند استخدام الخاصية FirstName، وكذلك بند Set والذي يُحدد ماذا يَحدث إذا قام المبرمج بتغيير قيمة الخاصية FirstName وفي نهاية كود الخاصية توجد الجملة End Property والتي توضح انتهاء كود الخاصية.

٣- نقوم الآن بتعبئة بقية الكود التابع للخاصية FirstName كالتالي:

```
Public Property FirstName() As String
    Get
        Return Name1
    End Get
    Set(ByVal value As String)
        Name1 = value
    End Set
End Property
```

مع ملاحظة بأننا نقوم بكتابة الكود المظلل بالأصفر فقط. وكما وضحنا من قبل فـ Get تعني ماذا سوف يشاهد المبرمج عند اختياره للخاصية، و Set تعني ماذا يحدث عند تغيير الخاصية، وبنفس الطريقة نقوم بكتابة خاصية ثانية LastName كالتالي فبعد End Property نضغط Enter لنكتب الخاصية الجديدة كالتالي: (نفس الخاصية السابقة عدا التغييرات المظللة بالأصفر)

```
Public Property LastName() As String
    Get
        Return Name2
    End Get
    Set(ByVal value As String)
        Name2 = value
    End Set
End Property
```

أكملنا الآن تعريف المتغيرات ثم كتابة الخصائص، ننتقل الآن إلى الطرق Methods. سنقوم بإنشاء الطريقة Age والتي تحسب لنا عمر الموظف بمجرد إدخال تاريخ ميلاده.

٤- إنشاء الطُرق Methods

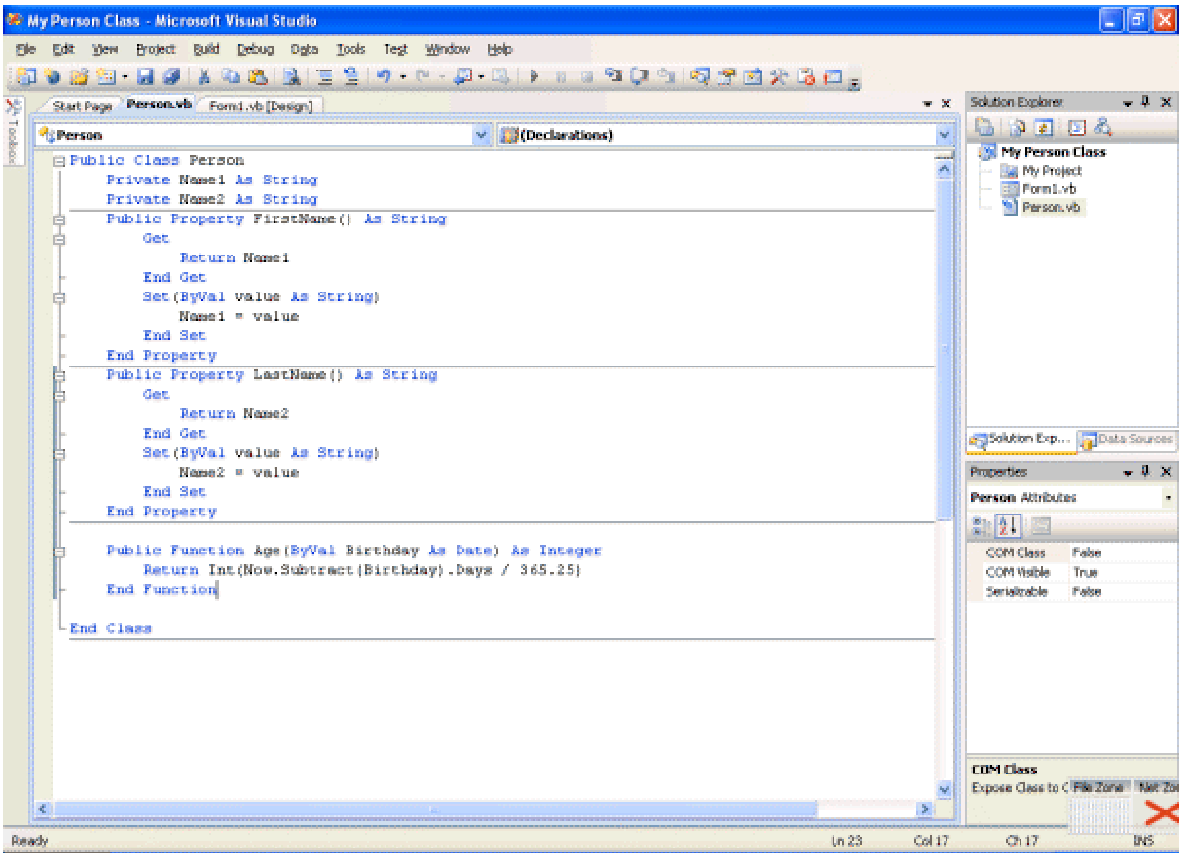
تحت الخاصية LastName نكتب تعريف الدالة Age كالتالي:

```
Public Function Age(ByVal Birthday As Date) As Integer
Return Int(Now.Subtract(Birthday).Days / 365.25)
End Function
```

لإنشاء طريقة Method في الفئة Class لهدف مُحدد، لابد من تعريف دالة Function أو إجراء مصغر Sub Procedure بداخل الفئة. ومعظم الطرق لا تحتاج إلى معاملات حتى تعمل، لكن الطريقة Age التي عرفناها تحتاج لـــ Birthday (وتعني تاريخ الميلاد الذي قمنا بتعريفه على أنه تاريخ) كمعامل وقمنا بتعريفه لِنُسنِد إليه تاريخ ميلاد الموظف ولتقوم الدالة بحساب عدد الأيام من تاريخ الميلاد إلى اليوم ثم قسمة المجموع على ٣٦٥,٢٥ (لإن عدد أيام السنة يساوي ٣٦٥ والسنوات الكبيسة (رابع سنة) تساوي ٣٦٦ لذلك قام المؤلف بالقسمة على ٣٦٥,٢٥ واستخدم Int لتقريب الكسر لأقرب رقم). لاحظ الدالة بين القوسين Now.Subtract(Birthday) حيث يقوم البرنامج بطرح تاريخ

الميلاد من التاريخ الحالي ثم تتم القسمة فيما بعد على ٣٦٥,٢٥ ثم بعدها تتم عملية التقريب. لمعرفة أكثر عن الدوال راجع الفصل العاشر من هذا الكتاب.

ملاحظة: التقويم الميلادي الذي يؤرخ لميلاد نبي الله عيسى عليه السلام يحتوي على سنوات كبيسة وغير كبيسة فالسنوات الكبيسة تحتوي على ٣٦٦ يوم وبقية السنوات تحتوي على ٣٦٥ يوم، ويبدأ التاريخ بثلاث سنوات غير كبيسة ثم سنة رابعة كبيسة، وللعلم تمت إضافة اليوم الـــ ٣٦٦ مؤخراً بسبب اعتماد التقويم الميلادي على دوران الأرض حول الشمس، ويتم إضافة اليوم ٣٦٦ كل رابع سنة لتتساوى السنة مع دورة الأرض حول الشمس، حيث تدور الأرض حول الشمس كل ٣٦٥ يوم وربع اليوم. ولمعرفة إذا كانت السنة كبيسة أو غير كبيسة نقسم رقم السنة على الرقم ٤ فإذا كان الناتج عدد صحيح بدون باقي كانت السنة كبيسة وإذا كان هناك باقي فالسنة غير كبيسة (خرجنا عن موضوع الكتاب). لاحظ شاشة بيئة التطوير بعد كتابة الفئة Person.



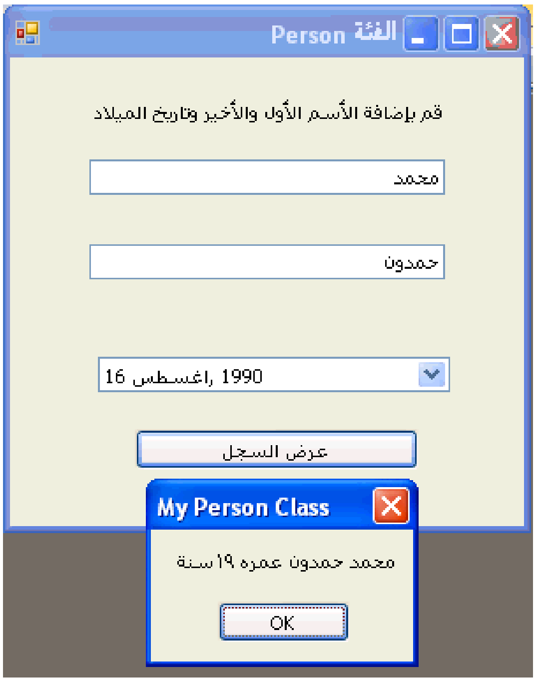
الآن نذهب إلى النموذج Form1 لاستخدام الفئة الجديدة في تطبيقنا.

ننقر نقرتين على الزر Button1 ونكتب الكود التالي في الحدث Click التابع للزر:

```
Dim Employee As New Person
Dim DOB As Date
Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
DOB = DateTimePicker1.Value.Date
MsgBox(Employee.FirstName & " " & Employee.LastName _
& " عمره " & Employee.Age(DOB) & " سنة")
```

نتذكر الآن ما قمنا به عند كتابة الفئة Person، ثم نطبق ما كتبناه بها على الكود أعلاه، قمنا بإسناد النص في مربع النص TextBox1 إلى المتغير FirstName ثم قمنا بإسناد النص في مربع النص الثاني TextBox2 إلى المتغير LastName، ثم قمنا بسحب التاريخ في DateTimePicker1 إلى المتغير DOB ، بعدها قمنا بكتابة كود لإظهار صندوق حوار يحتوي على الاسم الأول والأخير ثم العمر بالاعتماد على الفئة Person وبالأخص على الدالة Age الموجودة في الفئة المذكورة، حيث قمنا بإضافة التاريخ الذي قام باختياره المستخدم إلى المتغير DOB ثم وضعناه بدل المتغير BirthDay الموجود في الدالة Age الموجودة في الفئة Person، ليس في الأمر تعقيداً قم بقراءة الكود مرة ثانية إذا أحسست ببعض الصعوبة.

نقوم الآن بتنفيذ البرنامج ثم نكتب اسم في الاسم الأول وفي الاسم الأخير، ثم نحدد عمر تاريخ معين، ونضغط على الزر "عرض السجل". نلاحظ قيام البرنامج بكتابة الاسم الأول والأخير في صندوق الحوار الذي سيظهر وحساب العمر بالاعتماد على التاريخ الذي اخترناه، انظر الصورة، ملاحظة نسخة من المشروع موجودة في المرفق رقم ٠٤٩.



خطوة إضافية للأمام: وراثة فئة أولية Inheriting a Base Class

ما نستفيد من الوراثة هو أننا لا نقوم بكتابة الكود أكثر من مرة، حيث أننا نقوم بكتابة الكود مرة واحدة ثم نستفيد من الكود في كل مرة قد نحتاج له، وهذا هو المميز في البرمجة حيث أننا لا نقوم بكتابة الأكواد بشكل ممل، عند وراثة فئة أولية أو فئة قد تم تصميمها من قبل نستطيع إجراء بعض التعديلات لتتلاءم مع احتياجاتنا، لنأخذ مثال عن الفئة Person التي قمنا بتصميمها سابقاً وكيف سنقوم من بالاستفادة منها في فئة جديدة تتعلق بالمعلمين كما في المثال التالي:

١- نقوم بفتح الفئة Person التي قمنا بتصميمها سابقاً ونذهب إلى نهاية الكود بعد End Class لنضيف فئة جديدة، حيث بإمكاننا إضافة فئة جديدة بداخل نفس الملف أهم شيء أن تبدأ وتنتهي كل فئة بالأكواد Public Class و End Class، حيث نكتب بعد End Class الكود التالي للفئة الجديدة التي سنسميها Teacher:

```
Public Class Teacher
    Inherits Person
    Private Level As Short
```

```
    Public Property Grade() As Short
        Get
            Return Level
        End Get
        Set(ByVal value As Short)
            Level = value
        End Set
    End Property
End Class
```

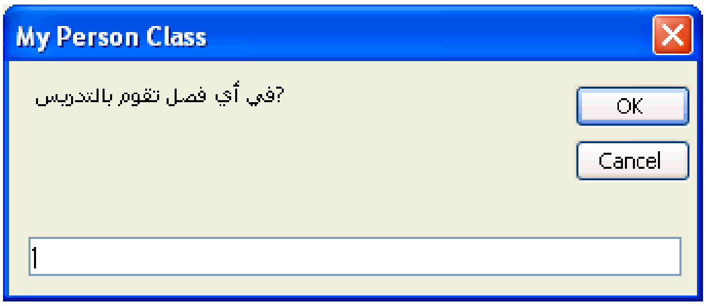
مع العلم بأننا سنقوم بطباعة الكود المظلل باللون الأصفر فقط، لاحظ بأننا قمنا بوراثة الفئة السابقة Person منذ السطر الأول من الكود، ماذا يعني ذلك: يعني بأن الفئة الجديدة ترث المتغيرات والخصائص والطرق Methods من الفئة السابقة Person وإذا كانت الفئة السابقة Person توجد في وحدة برمجية أُخرى Module أو في مشروع آخر فسيجب علينا تحديد مكانها باستخدام مجال الأسماء Namespace وباستخدام العبارة Imports التي نقوم بكتابتها في أول منطقة الكود، هل تتذكر متى قمت بكتابتها آخر مرة، في تلك المرة كنت تقوم بعملية وراثة لفئة من فئات الدوت نت، الآن الفئة Teacher تحتوي على نفس الخصائص الموجودة في الفئة Person وهما FirstName و LastName، بالإضافة إلى الخاصية الجديدة التي أضفناها في الفئة Teacher وهي الخاصية Grade والتي تسجل الفصل الذي يقوم المدرس بالتدريس فيه، الآن لنذهب إلى النموذج Form1 ونستعرض الكود الخاص بالزر Button1 ونقوم بتعديل الكود كالتالي:

```
Dim Employee As New Teacher
Dim DOB As Date
Employee.FirstName = TextBox1.Text
Employee.LastName = TextBox2.Text
```

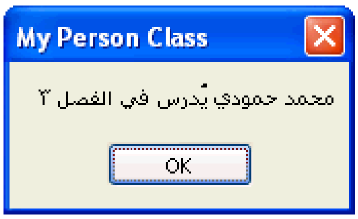
```
DOB = DateTimePicker1.Value.Date
Employee.Grade = InputBox("في أي فصل تقوم بالتدريس?")
MsgBox(Employee.FirstName & " " & Employee.LastName _
& " يُدرس في الفصل " & Employee.Grade)
```

ملاحظة أننا قمنا بتغيير ما هو مظلل باللون الأصفر فقط، قمنا بتغيير الفئة من Person إلى Teacher حيث أنها نفس الفئة (لأن Teacher ترث من Person) مع إضافة الخاصية Grade لها فقط. وتم استبعاد عملية حساب العمر لأننا لا نحتاج لها من هذا نستفيد بأننا إذا لم نحتاج لخاصية معينة ضمن الفئة فليس بالضرورة أن نستخدمها وإنما نستخدم ما نحتاج فقط من الخصائص من الفئة التي نقوم بالاستفادة منها.

نقوم الآن بتنفيذ الكود وبعد تعبئة الاسم الأول والأخير والضغط على الزر "عرض السجل" يظهر صندوق الإدخال:



نقوم بكتابة رقم وليكن الرقم ٣ في صندوق الإدخال ثم نضغط Ok. سنشاهد الشاشة:



ملاحظة: عند صندوق الإدخال نقوم بكتابة أرقام وليس حروف لأننا عرفنا الخاصية Grade على أساس أنها من النوع Short (والذي يبدأ من 32.768– إلى 32.767). نسخة من المثال السابق في المرفق رقم ٠٥٠.

إذا أعجبك هذا الفصل المتضمن معلومات عن الوراثة والبرمجة كائنية التوجه، أستطيع أن أقول لك بأن لغة الفيجوال بيسك ٢٠٠٨ هي لغة كائنية التوجه بحق، تستطيع إضافة أحداث للفئات وكذلك تستطيع إضافة قِيم أولية للخصائص Defaults، وتستطيع استخدام anonymous types و overloading. هذه الميزات وغيرها التابعة للبرمجة كائنية التوجه تستطيع معرفتها بقراءة التعليمات المرفقة مع برنامج الفيجوال بيسك، أو بقراءة كتاب متخصص.

خلاصة الفصل السادس عشر

من اجل أن	قم بالتالي
وراثة نموذج موجود	من القوائم نختار Project ثم Add New Item ثم نختار Inherited Form ثم نحدد اسم للنموذج الجديد ثم نختار Add، من النافذة التي تظهر نختار النموذج (الفورم) الذي نريد وراثته ثم نختار Ok، مع ملاحظة أننا إذا أردنا وراثة نموذج (فورم) ليس ضمن ملفات مشروعنا فلابد أن يكون بصيغة dll.
تعديل النموذج الابن	نستطيع تعديل النموذج الابن (الذي قمنا بوراثته من نموذج سابق) وذلك بتكبيره وتصغيره أو بإضافة مكونات جديدة، وتعديل خصائص المكونات، فقط لا نستطيع تعديل المكونات الموجودة فيه من قبل.
إضافة فئات خاصة بك إلى المشروع	من القوائم نختار Project ثم Add New Item ثم نختار Class ونقوم بتحديد اسم للفئة ثم نختار Add، بعدها نقوم بكتابة كود للفئة المضافة.
إخفاء المتغيرات المعرفة بداخل الفئة	لإخفاء المتغيرات التي قمنا بتعريفها بداخل الفئة (حتى لا يقوم المبرمجين بالاستفادة من المتغيرات بدون استخدام الفئة) نستخدم كلمة التعريف Private بداخل الفئة كالتالي:

	Private Name1 As String
إنشاء خاصية جديدة بداخل الفئة	نقوم بتعريف خاصية عامة بداخل الفئة كالتالي: Public Property FirstName() As String Get Return Name1 End Get Set(ByVal value As String) Name1 = value End Set End Property
إنشاء طريقة Method جديدة بداخل الفئة	نقوم بتعريف دالة أو إجراء مصغر بداخل الفئة كالتالي: Public Function Age(ByVal Birthday As Date) As Integer Return Int(Now.Subtract(Birthday).Days / 365.25) End Function
تعريف متغير لاستخدام الفئة	نستخدم كلمة التعريف Dim ثم الكلمة New ثم نذكر اسم الفئة التي نريد الاستفادة منها كالتالي: Dim Employee As New Person
تعيين الخصائص للمتغير	نستخدم الطريقة العامة لتعيين الخصائص كالتالي: Employee.FirstName = TextBox1.Text

وراثة فئة في فئة أخرى، (وراثة الفئة الأب في الفئة الإبن)	نقوم بإنشاء فئة جديدة ونستخدم الكلمة Inherits لوراثة الفئة الأب كالتالي: Public Class Teacher Inherits Person Private Level As Short Public Property Grade() As Short Get Return Level End Get Set(ByVal value As Short) Level = value End Set End Property End Class

الفصل السابع عشر: التعامل مع الطابعات

يوجد في فيجوال بيسك ٢٠٠٨ فئة تسمى PrintDocument تستخدم لعملية الطباعة، ونستطيع طباعة نصوص أو رسومات أو غيره باستخدام الفيجوال بيسك ٢٠٠٨. دعونا نتعرف أكثر على الفئة PrintDocument.

استخدام الفئة PrintDocument

معظم البرامج تحت نظام الويندوز تسمح لنا بطباعة المستندات والملفات التي نتعامل معها، وتم دعم عملية الطباعة ضمن برنامج الفيجوال بيسك ٢٠٠٨ بالمقارنة مع نسخة الفيجوال بيسك ٦.

فباستخدام الفئة PrintDocument يمكننا الطباعة بشكل أمثل، حيث يمكننا استخدام الفئة PrintDocument بطريقتين وهما:

- إضافة المكون PrintDocument إلى النموذج (الفورم).
- تعريف الفئة PrintDocument بواسطة الكود في مرحلة الكود.

توجد الفئة PrintDocument ضمن مجال الأسماء System.Drawing.Printing والذي يحتوي على العديد من الميزات الهامة لطباعة النصوص والرسومات من ضمنها الكائن PrinterSetting والذي يقوم بتغيير إعدادات الطابعة، وكذلك الكائن PageSettings ويحتوي على إعدادات الصفحات ويمكن تغييرها منه، والكائن PrintPageEventArgs الذي يحتوي علة معلومات عن الصفحة التي نقوم بطباعتها، ويضاف مجال الأسماء System.Drawing.Printing بشكل مباشر لتطبيقك وإذا لم يُضاف أو أردنا إضافته بشكل مباشر حتى يتسنى لنا التعامل مع مكوناته بسهولة نضيف الكود التالي إلى أول الكود:

```
Imports System.Drawing.Printing
```

لنأخذ الآن مثال على استخدام المكون PrintDocument لطباعة ملف رسومات من النظام:

١-نقوم بإنشاء مشروع جديد باسم My Print Graphics

٢-نقوم بإضافة ليبل وصندوق نص وكذلك زر بالإضافة للمكون PrintDocument

٣-نقوم ببعض التعديلات على المكونات وعلى النموذج حتى تصبح المكونات كما في الصورة التالية:



٤- الآن سنقوم بإضافة بعض الكود الذي يسمح لنا بطباعة الصورة بمختلف أنواعها (على هيئة JPEG bitmap, icon, metafile,)، فقط نقوم بالنقر مرتين Double-Click على الزر "طباعة الصورة"، ليقوم الفيجوال بيسك بتحويلنا مباشرة إلى منطقة الكود وإلى الحدث Click التابع للزر "طباعة الصورة"، لكننا سنذهب إلى الأعلى في منطقة الكود إلى أعلى منطقة الكود لكتابة كود استيراد مجال الأسماء System.Drawing.Printing حيث نكتب التالي في أعلى منطقة الكود:

```
Imports System.Drawing.Printing
```

الكود أعلاه يسهل لنا عملية استدعاء فئات الطباعة ضمن الكود.

٥- نعود الآن إلى الحدث Click التابع للزر "طباعة الصورة" ونكتب الكود التالي:

```
' الطباعة مع صيد الأخطاء إن وُجدت
Try
    AddHandler PrintDocument1.PrintPage, AddressOf
Me.PrintGraphic
    PrintDocument1.Print() 'لطباعة الصور
```

```
Catch ex As Exception 'مراقبة فيما إذا كان هناك خطأ
    MessageBox.Show("عفواً -- هناك خطأ ما", ex.ToString())
End Try
```

بعد كتابة الكود ستلاحظ خط أزرق متعرج تحت الكلمة Me.PrintGraphic تجاهله لأننا سنقوم بتعريفه في الخطوة التالية، استخدمنا الجملة AddHandler والتي تقوم باختيار الحدث المناسب لتستدعيه لحظة تشغيل الحدث PrintPage، مع ملاحظة أن AddHandler ليس له علاقة بمراقبة الأخطاء وإنما يقوم بمتابعة أحداث الكائن، بعدها قمنا بإضافة كود صيد الأخطاء باستخدام الكلمة Catch بنفس الطريقة التي استخدمناها في الفصل التاسع من هذا الكتاب مع فارق بسيط يسمح للمتغير Ex أن يجمع لنا معلومات مفصلة عن الخطأ إن وُجِد.

نخرج الآن من حدث Click ونكتب الكود التالي فوقه مباشرة:

```
'إجراء لطباعة الرسومات
Private Sub PrintGraphic(ByVal sender As Object, _
ByVal ev As PrintPageEventArgs)
    'نقوم بإنشاء الرسم باستخدام DrawImage
    ev.Graphics.DrawImage(Image.FromFile(TextBox1.Text), _
    ev.Graphics.VisibleClipBounds)
    ' قم بالتأكيد أن هذه آخر صفحة تقوم بطباعتها
    ev.HasMorePages = False
End Sub
```

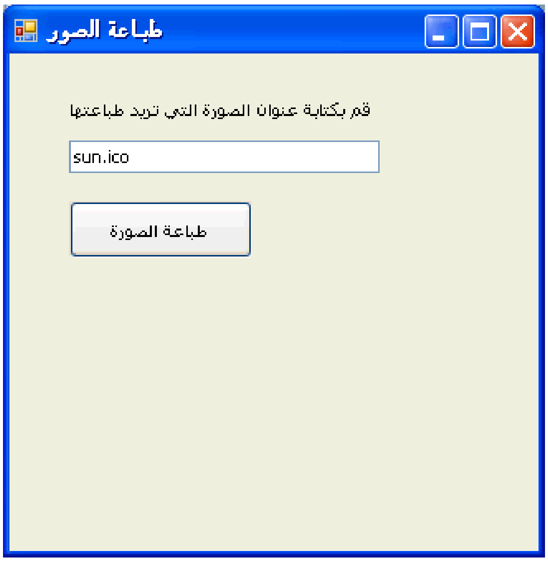
هذا الإجراء يتبع المكون PrintDocument1 وتقوم الطريقة PrintDocument1.Print بإنشائه، قمنا هنا بإلحاق المتغير ev بـــ PrintPageEventArgs لأنه يتبع

PrintGraphic ويحتوي هذا المتغير على معلومات عن الصفحة التي نريد طباعتها، طبعاً كل هذه المتغيرات والإجراءات تتبع مجال الأسماء System.Drawing.Printing.

استخدمنا في الكود السابق الطريقة Graphics.DrawImage لطباعة الصورة التي نأخذها من ملف وصلة الملف موجودة بداخل صندوق النص، تذكر بأننا كتبنا في صندوق النص Sun.ico وعلية يجب أن تكون الصورة Sun.ico مع البرنامج في نفس الملف (أي في ملف bin\Debug) ليتم طباعتها وإذا كانت في ملف آخر فيجب تحديد مسار الصورة الجديد ليتم طباعتها وإلا سيظهر لنا خطأ. في نهاية الكود قمنا بضبط ev.HasMorePages على False لتحديد أنه لا توجد صفحات إضافية لطباعتها.

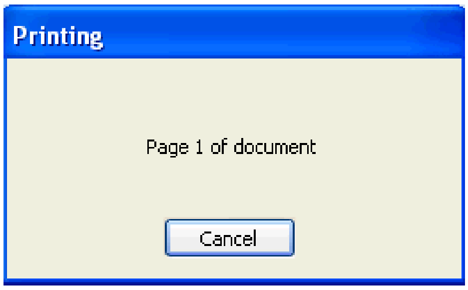
قم بتنفيذ البرنامج الآن ثم قم بتحديد مكان الصورة وقم بطباعتها، ملاحظة: توجد نسخة من المشروع في المرفق رقم ٠٥١.

بعد تنفيذ المشروع تظهر هذه النافذة:



نقوم بتحديد مكان الصورة الصحيح، ولأن الصورة Sun.ico مرفقة في ملف bin\Debug نعتمد المسار (أما إذا كانت الصورة التي تريد طباعتها موجودة في ملف آخر فنقوم باختيار مسار الصورة الجديدة ووضعه في مربع النص) ثم نضغط الزر "طباعة الصورة" (مع ملاحظة

بأن الطريقة DrawImage تقوم بتكبير الصورة التي نقوم بطباعتها حتى تكون ملئ الصفحة)، ستظهر نافذة الطباعة كالتالي:



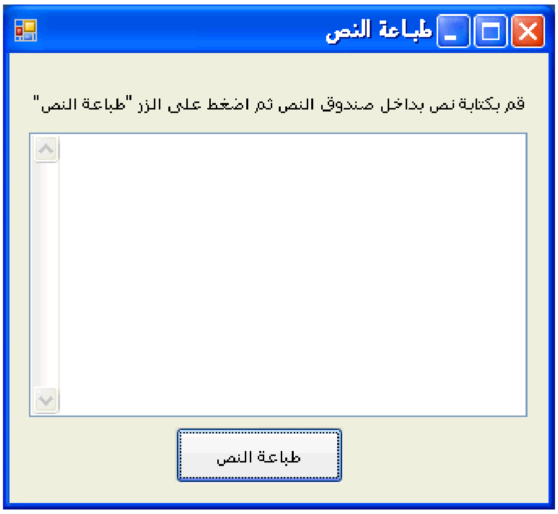
تستطيع طباعة أكثر من صورة بتحديد مسار كل صورة ثم طباعتها.

طباعة النص من مربع النص

تعلمنا في المثال السابق كيفية طباعة صورة من جهازك على الطابعة الرئيسية للجهاز، سنتعلم الآن كيف نقوم بطباعة نص على الطابعة الرئيسية للجهاز باستخدام نفس الفئة PrintDocument لكننا لن نقوم بإضافة مكون الطباعة من نافذة الأدوات سنقوم بتعريف الفئة PrintDocument بواسطة الكود، وسنستخدم الطريقة Graphics.DrawString لطباعة النص من صندوق النص، مع ملاحظة بأننا في هذا المثال سنتعلم كيف نطبع صفحة واحدة من النصوص أو أقل أما إذا أردنا طباعة أكثر فيجب علينا كتابة أكواد أكثر وهذا ما سنتعلمه لاحقاً، لكن دعونا نتعلم عملية الطباعة خطوة خطوة.

١- ننشئ مشروع جديد باسم My Print Text

٢-نضيف ليبل وصندوق نص وزر للنموذج ونقوم بالتعديلات اللازمة ليصبح النموذج كما ف الصورة التالية:



٣-مرحلة الكود ننقر نقرتين Double-Click على الزر طباعة النص ونذهب إلى أعلى الكود لنقوم باستيراد مجال الأسماء كالتالي:

```
Imports System.Drawing.Printing
```

كما وضحنا سابقاً بان الكود أعلاه يسهل علينا عملية استدعاء فئات وطُرق الطباعة في مرحلة الكود، نذهب الآن إلى الحدث Click التابع للزر "طباعة النص" ونكتب الكود:

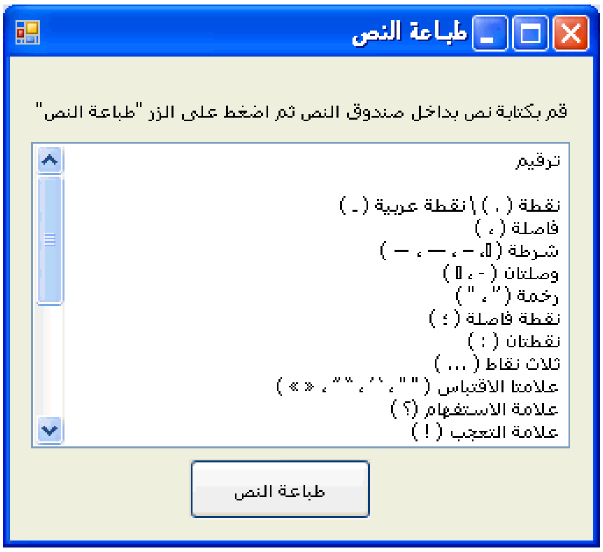
```
' الطباعة مع صيد الأخطاء
Try
    ' نعرف المتغير PrintDoc على أنه من النوعية PrintDocument
    Dim PrintDoc As New PrintDocument
    AddHandler PrintDoc.PrintPage, AddressOf Me.PrintText
    PrintDoc.Print() 'لطباعة النص
Catch ex As Exception 'صيد الأخطاء
    MessageBox.Show("عفواً--حدث خطأ عند عملية الطباعة",
ex.ToString())
End Try
```

مع ملاحظة تشابه هذا الكود مع كود طباعة الصور والفرق ما هو مظلل بالأصفر فقط، الآن بدلاً من إضافة المكون PrintDocument سنقوم بإضافته بشكل برمجي بواسطة الكود، نقوم الآن بكتابة إجراء الطباعة أعلى كود الحدث Click كالتالي:

```
'إجراء لطباعة النصوص
Private Sub PrintText(ByVal sender As Object, _
ByVal ev As PrintPageEventArgs)
    'للتعبير عن النصوص بشكل رسومي DrawString نستخدم الطريقة
    ev.Graphics.DrawString(TextBox1.Text, New Font("Arial", _
    11, FontStyle.Regular), Brushes.Black, 120, 120)
    ' نقوم بتحديد أن هذه الصفحة الأخير التي نطبعُها
    ev.HasMorePages = False
End Sub
```

نلاحظ تشابه الكود مع كود طباعة الصور مع الفرق الملون باللون الأصفر، في طباعة النصوص نستخدم الطريقة DrawString بدلاً من الطريقة DrawImage التي تستخدم لطباعة الصور، مع العلم بأن الطريقة DrawString لا تقوم بتكبير النصوص لتملئ الصفحة ولكنها تعتمد على نوعية الخط ومقاسه التي قمنا بتحديدها كما هو موضح في الكود. كما قمنا بتحديد النقطة العمودية والأفقية على صفحة الطباعة (120, 120) التي سيقوم البرنامج بالبدء من عندها في عملية الطباعة، ليظهر النص المطبوع كما هو في صندوق النص، كما قمنا في نهاية الكود بتحديد أن هذه هي نهاية الصفحة التي يقوم البرنامج بطباعتها (أي أن النص من صفحة واحدة وليس من صفحتين أو أكثر) كما في المثال السابق.

بعد إضافة الكود قم بتنفيذ البرنامج وقم بطباعة بعض النصوص في صندوق النص، ستكون نافذة البرنامج كالتالي:



بعد كتابة النص في صندوق النص نضغط على الزر "طباعة النص"، فسيقوم البرنامج مباشرة بطباعة النص على طابعتك الافتراضية.

ملاحظة: لم يتم تدعيم التفاف النص حتى الآن في الطباعة، فإذا كان السطر طويلاً فلن يقوم البرنامج بطباعته في أكثر من سطر (جرب كتابة سطر طويل في صندوق النص(بدون الضغط على الزر Enter))، سنحل هذه المشكلة لاحقاً وكما قلنا بأننا نتعلم عملية الطباعة خطوة خطوة، كما أن البرنامج لا يطبع من اليمين إلى اليسار وإنما من اليسار إلى اليمين هذه ستحتاج مراجعة في التعليمات المرفقة مع برنامج الفيجوال بيسك Documentation لتجاوزها.

نسخة من المشروع في المرفق رقم ٠٥٢.

طباعة ملفات نصية من أكثر من صفحة

تعلمنا إلى الآن عمليات الطباعة البسيطة للنصوص والصور، في حالة التعامل مع ملفات النصوص الكبيرة التي تحتوي على أكثر من صفحة فإن الأمر يختلف، فالعيوب التي تترافق مع الطرق السابقة هي: لا يمكن طباعة الأسطر الطويلة، لا يمكن طباعة أكثر من صفحة، لماذا لا يتم طباعة أكثر من صفحة لإن طريقة الطباعة لا تفهم معنى الصفحة فقط تقوم بطباعة النص المطلوب طباعته في الطابعة إذا كان النص أكثر من صفحة فستقوم بطباعة الصفحة الأولى فقط وعدم الاهتمام إذا كان هناك أكثر من صفحة، ولحل هذه الإشكالية لابد من إنشاء صفحة وهمية (صفحة افتراضية) تسمى PrintPage ثم إرسال النص إليها حتى تمتلئ ومنها

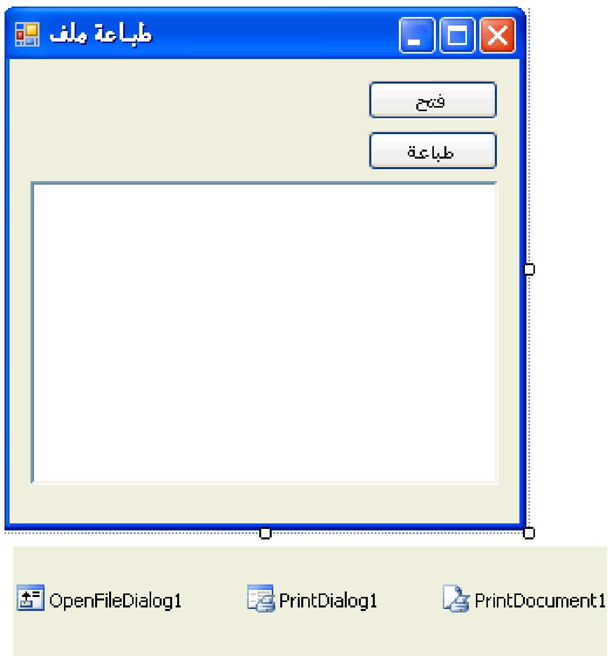
نرسل الصفحة كاملة إلى الطابعة ويتم إعادة هذا الإجراء حتى تنتهي الصفحات التي نقوم بطباعتها، هناك طريقة أُخرى لطباعة النص متعدد الصفحات باستخدام الطريقة Graphics.MeasureString حيث تقوم هذه الطريقة بحساب عدد الأحرف اللازم طباعتها ونوع الخط وحجم الحروف ثم حساب مقاسات الصفحة ثم تنفيذ الطباعة حتى آخر صفحة.

لنأخذ الآن مثال على طباعة أكثر من صفحة والتعامل مع أدوات التحكم المتقدمة:

١- نقوم بإنشاء مشروع باسم My Print File

٢- نقوم بإضافة اثنين أزرار وصندوق نص من النوع المتقدم RichTextBox وكذلك المكون OpenFileDialog ثم نضيف المكونات التالية للطباعة PrintDocument و PrintDialog . المكون الأخير PrintDialog يستخدم لعرض خيارات الطباعة في برنامجك قبل طباعة أي صفحة.

٣- نقوم بتغيير الخاصية Name للزر الأول إلى btnOpen وللثاني إلى btnPrint ونغير الخاصية Enabled للزر الثاني إلى False ونقوم بتغيير بعض الخصائص للكائنات على النموذج حتى يصبح النموذج كما في الشكل:



٤-ننقر نقرتين Double-Click على الزر "فتح" لنذهب إلى الحدث Click التابع للزر في منطقة الكود، وفي منطقة الكود نذهب لأعلى الكود ونكتب الكود التالي لاستيراد مجال الأسماء:

```
Imports System.IO 'for FileStream class ما يسمى بتدفق الملف
Imports System.Drawing.Printing
```

تم استيراد مجال الأسماء الأول System.IO لقراءة الملف النصي بطريقة ما يسمى بالتدفق FileStream (لا أعرف فيما إذا كانت الترجمة صحيحة)، فطريقة التدفق هي قراءة الملف النصي من الأول إلى الأخير بشكل متسلسل، بحيث تقرأ كل سطر على حده أو كل كلمة على حده.

٥-نقوم بتعريف المتغيرات العامة تحت Public Class Form1 نكتب هذا الكود:

```
Private PrintPageSettings As New PageSettings
Private StringToPrint As String
Private PrintFont As New Font("Arial", 10)
```

تقوم هذه التعريفات بتحديد معلومات عن الصفحات التي سنقوم بطباعتها.

٦-نذهب الآن للحدث btnOpen_Click لنكتب الكود التالي:

```
Dim FilePath As String
'عرض نافذة حوار للبحث عن ملف نصي
OpenFileDialog1.Filter = "Text files (*.txt)|*.txt"
OpenFileDialog1.ShowDialog()
'إذا لم يختر المستخدم "إلغاء الأمر"، قم بتحميل مسار الملف
If OpenFileDialog1.FileName <> "" Then
    FilePath = OpenFileDialog1.FileName
    Try
```

```
            'قم بقراءة الملف النصي وتحميله في صندوق النص
            Dim MyFileStream As New FileStream(FilePath,
FileMode.Open)
            RichTextBox1.LoadFile(MyFileStream, _
            RichTextBoxStreamType.PlainText)
            MyFileStream.Close()
            'تحديد النصوص للطباعة
            StringToPrint = RichTextBox1.Text
            'تفعيل الزر طباعة
            btnPrint.Enabled = True
        Catch ex As Exception
            'عرض رسالة الخطأ إذا حدث خطأ
            MessageBox.Show(ex.Message)
        End Try
    End If
```

عندما يقوم المستخدم بالضغط على الزر "فتح" سيقوم البرنامج بفتح نافذة حوار له ليختار ملف، وسيضع البرنامج فلتر بحيث لا تظهر إلا الملفات النصية باعتماد اللاحقة Txt في الفلتر، إذا لم يلغي المستخدم الأمر وقام باختيار ملف نصي فسيقوم البرنامج بقراءة الملف النصي وكتابته في صندوق النص، وسيقوم بتفعيل الزر "طباعة" ليستطيع طباعة النص، بقي لنا الآن كود الطباعة وكود إظهار خيارات الطباعة.

٧-بالنقر مرتين على الزر "طباعة" سينتقل البرنامج مباشرة إلى منطقة الكود وإلى الحدث Click التابع للزر نكتب الكود التالي:

```
    Try
```

```
'حدد خيارات الصفحة الحالية
PrintDocument1.DefaultPageSettings = PrintPageSettings
'قم بتحديد النص المراد طباعته وقم بإظهار خيارات الطباعة
StringToPrint = RichTextBox1.Text
PrintDialog1.Document = PrintDocument1
Dim result As DialogResult = PrintDialog1.ShowDialog()
'قم بطباعة الصفحة إذا ضغط المستخدم على نعم أو Ok
If result = DialogResult.OK Then
    PrintDocument1.Print()
End If
Catch ex As Exception
'لعرض رسالة خطأ إذا حدث
MessageBox.Show(ex.Message)
End Try
```

في الكود أعلاه قمنا بتحديد خيارات الطباعة الافتراضية، كما قمنا بإسناد النص الموجود بداخل صندوق النص إلى المتغير StringToPrint الذي تم تعريفه في بداية صفحة الكود، بعدها يقوم الكود بعرض خيارات الطباعة ليقوم المستخدم باختيار الطابعة المراد الطباعة فيها وكذلك عدد النسخ أو الطباعة إلى ملف وغيرها من الخيارات، إذا قام المستخدم بالموافقة وذلك بالضغط على Ok أو مرافق يقوم البرنامج بإرسال أمر الطباعة إلى الطابعة بالكود:

```
PrintDocument1.Print()
```

سنقوم الآن بكتابة الكود اللازم للمكون PrintDocument1.

٨-نذهب إلى منطقة التصميم وننقر نقرتين Double-Click على المكون PrintDocument1 ليأخذنا الفيجوال بيسك مباشرة إلى الحدث PrintPage التابع للمكون نكتب الكود التالي في الحدث:

```
Dim numChars As Integer
Dim numLines As Integer
Dim stringForPage As String
Dim strFormat As New StringFormat
'بالإعتماد على إعدادات الصفحة، قم بتحديد مستطيل وهمي على الصفحة
Dim rectDraw As New RectangleF( _
e.MarginBounds.Left, e.MarginBounds.Top, _
e.MarginBounds.Width, e.MarginBounds.Height)
'قم بتعريف مساحة لتحديد كمية النصوص اللازم طباعتها على الصفحة الواحدة
'قم بإعتماد طول الصفحة ناقصاً سطر واحد حتى نطبع النص بصورة صحيحة
Dim sizeMeasure As New SizeF(e.MarginBounds.Width, _
e.MarginBounds.Height - PrintFont.GetHeight(e.Graphics))
'عند طباعة النصوص الطويلة، فارق بين الكلمات
strFormat.Trimming = StringTrimming.Word
'Compute how many chars and lines can fit based on sizeMeasure
' قم بحساب كم عدد الأسطر والحروف بالإعتماد على المقاسات
e.Graphics.MeasureString(StringToPrint, PrintFont, _
sizeMeasure, strFormat, numChars, numLines)
```

```
'قم بحساب عدد الحروف التي تملئ الصفحة الواحدة
stringForPage = StringToPrint.Substring(0, numChars)
'قم بطباعة النص على الصفحة الحالية
e.Graphics.DrawString(stringForPage, PrintFont, _
Brushes.Black, rectDraw, strFormat)
'إذا كان هناك نص إضافي قم بالتوضيح بأن هناك صفحات إضافية
If numChars < StringToPrint.Length Then
    'قم بطرح النص المطبوع من بقية النص
    StringToPrint = StringToPrint.Substring(numChars)
    e.HasMorePages = True
Else
    e.HasMorePages = False
    'All text has been printed, so restore string
    ' بعد طباعة النص كاملاً، قم بإعادة النص إلى صندوق النص
    StringToPrint = RichTextBox1.Text
End If
```

ستلاحظ التعليقات المكتوبة عند كل سطر من الكود، وكما قلنا سابقاً التعليقات لا علاقة لها بالكود، وإنما تسهل علينا عملية مراجعة الكود تبدأ التعليقات بـــ ' في أول سطر الكود.

يقوم الكود أعلاه بحساب مستطيل الطباعة على الصفحة بالإعتماد على الإعدادات المختارة من نافذة خيارات الطباعة، بعد حساب مساحة مستطيل الطباعة يقوم بطباعة الأسطر وإذا كان هناك سطر طويل يقوم بعملية التفاف للنص إلى السطر الذي يليه حتى تتم طباعة كل النص بطريقة طبيعية، تم تعريف منطقة الطباعة بالمتغير rectDraw والذي يعتمد على الفئة RectangleF

وتم استخدام المتغير strFormat والطريقة Trimming لحذف النصوص خارج حدود الصفحة، تمت طباعة النص باستخدام طريقة طباعة النصوص DrawString، أما خيار الصفحات الإضافية HasMorePage فتم ضبطه على True لإن النص طويل ويحتوي على أكثر من صفحة، في الأخير تمت إعادة محتويات صندوق النص RichTextBox1 إليه.

صحيح أن هذا المشروع طويل ومعقد لكن صدقني هذه الطريقة ستستخدمها في أكثر مشاريع الطباعة تعقيداً ولن تحتاج أكثر من هذا، لذلك قم بمراجعة الكود أكثر من مرة للاستفادة والتعلم من هذا المشروع، بالنسبة للمستخدم العربي ستحتاج دالة أو طريقة معينة لتحويل النص من اليمين إلى اليسار وهذا ليس صعباً بالنسبة لمن سيبحث في تعليمات الفيجوال بيسك Documentation أو على إنترنت.

قم بتنفيذ المشروع وجرب الطباعة من ملف نصي كبير يحتوي على أكثر من صفحة، سيفتح لك البرنامج خيارات الطباعة لاختيار الطابعة المناسبة وبعد اختيار الطابعة والموافقة سيقوم البرنامج بطباعة النص في الملف النصي.

مبروك لقد قمت بتصميم برنامج الطباعة الذي قد تحتاج لتطبيقه في أي مهمة طباعة في مشاريعك.

ملاحظة: نسخة من المشروع في المرفق رقم ٠٥٣.

خطوة إضافية للأمام: إضافة معاينة لصفحة الطباعة وكذلك صفحة إعدادات

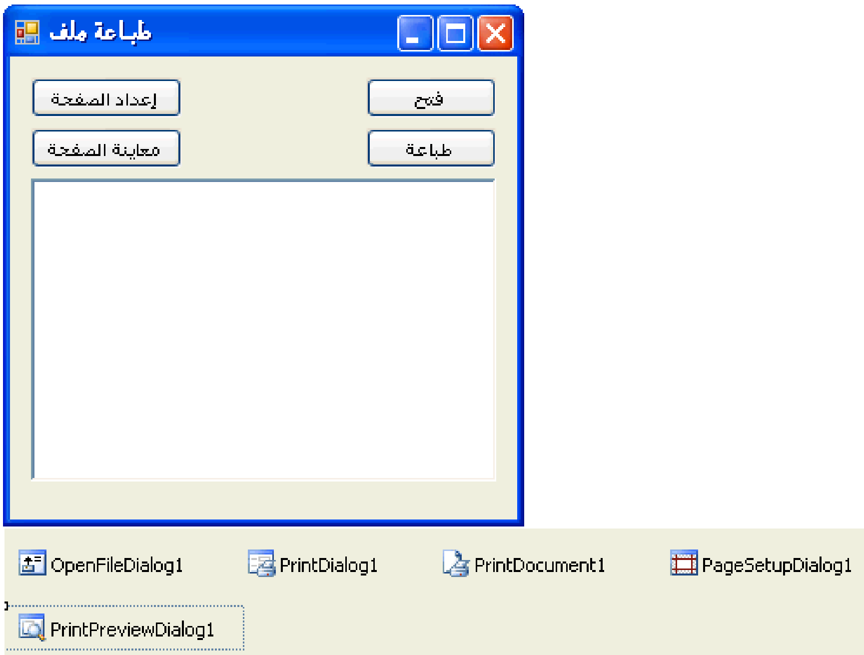
التطبيق السابق يشمل على العديد من مهام الطباعة، لكنك يمكنك أن تصبح أكثر احترافية بإضافة صفحة معاينة قبل الطباعة وكذلك صفحة لإعدادات صفحة الطباعة، المكون PrintPreviewDialog يستخدم لعرض صفحة معاينة قبل الطباعة، أما المكون PageSetupDialog فيستخدم لعرض إعدادات صفحة الطباعة، مثل بقية المكونات تستطيع إضافة هذين المكونين من صندوق الأدوات إلى مشاريعك أو تعريفهما وإضافتهما بواسطة الكود، سنستفيد من المشروع السابق ونبدأ:

١- إذا لم تقم بتطبيق المشروع السابق يمكنك الاستفادة من المرفق رقم ٠٥٣، نفتح المشروع السابق.

٢–نقوم بإضافة اثنين أزرار للنموذج (الفورم)، ثم نقوم بإضافة المكون PrintPreviewDialog وكذلك المكون PageSetupDialog إلى المشروع.

٣–نقوم ببعض التعديلات على خصائص الأزرار المضافة كالتالي: الزر الأول Name نغيره لـــ btnSetup والخاصية Text لـــ "إعداد الصفحة" أما الزر الثاني فـــ Name تغيرها إلى btnPreview و Text إلى "معاينة الصفحة" ونضبط الخاصية Enabled لكلاهما على False.

٤–بعد التعديلات سيظهر النموذج (الفورم) بهذا الشكل:



٥–ننقر نقرتين على الزر "إعداد الصفحة" ونكتب الكود التالي في الحدث Click التابع للزر:

```
Try
    'قم بتحميل إعدادات الصفحة وإظهار نافذة إعدادات الصفحة
    PageSetupDialog1.PageSettings = PrintPageSettings
    PageSetupDialog1.ShowDialog()
Catch ex As Exception
```

```
    'إظهار رسالة خطأ إن وُجِد
    MessageBox.Show(ex.Message)
End Try
```

الكود أعلاه سيقوم بإظهار نافذة أعدادات صفحة الطباعة، قد قمنا بتعريف المتغير PrintPageSettings في بداية صفحة الكود، هذا المتغير يحتوي على القيم الافتراضية لإعدادات القيمة ونستخدم المكون PageSetupDialog1 لفتح نافذة إعدادات الطباعة وأخذ القيم من هذا المتغير ثم التعديل على القيم في حالة تعديل بعض الإعدادات مثل هوامش الطباعة وغيرها.

٦- نقوم بإظهار النموذج مرة أُخرى والنقر مرتين Double-Click على الزر "معاينة الصفحة" ثم نكتب الكود التالي في منطقة الكود:

```
Try
    'قم بتحديد إعدادات الطباعة للصفحة الحالية
    PrintDocument1.DefaultPageSettings = PrintPageSettings
    'قم بتحديد مستند الطباعة ثم عرضه قبل الطباعة
    StringToPrint = RichTextBox1.Text
    PrintPreviewDialog1.Document = PrintDocument1
    PrintPreviewDialog1.ShowDialog()
Catch ex As Exception
    'إظهار رسالة خطأ إن وُجِد
    MessageBox.Show(ex.Message)
End Try
```

يقوم الكود أعلاه بأخذ الإعدادات الافتراضية، هذه الإعدادات هامة جداً حيث يستفيد منها البرنامج في حساب عدد الحروف في الصفحة الواحدة، ثم يقوم بأخذ النص من صندوق النص RichTextBox1 لغرض إظهاره في نافذة المعاينة قبل الطباعة.

٧- إذا قمنا بتنفيذ البرنامج الآن ستظهر لنا هذه الشاشة:



لعلنا سنتذكر الآن بأننا لم نكتب كود يقوم بتفعيل الأزرار "إعداد الصفحة" و "معاينة الصفحة" بعد تحميل الملف النصي في صندوق النص، نعود إلى الكود لنضيف كود يقوم بتفعيل الأزرار المذكورة بعد فتح ملف نصي في صندوق النص:

في الحدث Click التابع للزر "فتح" btnOpen نضيف كيف التفعيل بعد كود تفعيل الزر طباعة كالتالي: فبعد هذا السطر من الكود:

```
btnPrint.Enabled = True
```

نضيف هذين السطرين لتفعيل الزرين "إعداد الصفحة" و "معاينة الصفحة" على التوالي:

```
btnSetup.Enabled = True
btnPreview.Enabled = True
```

٨-نقوم الآن بتنفيذ البرنامج وفتح ملف نصي، ثم تعديل إعدادات صفحة الطباعة من الزر "إعداد الصفحة" ثم معاينة الصفحة قبل الطباعة، قم بالتعديل على الصفحة مرة ثانية وثالثة ثم مشاهدة أثر تعديلاتك على صفحة معاينة قبل الطباعة.

٩- ملاحظة نسخة من المشروع في المرفق رقم ٠٥٤.

قمت بتجربة المشروع الأخير وتطبيق بعض التعديلات ولاحظت السهولة الكبيرة في التعامل مع نافذة الطباعة وإعدادات الطباعة وكذلك المعاينة قبل الطباعة، وكلمة حق هذا المشروع من أفضل المشاريع في هذا الكتاب ويجيب على الكثير من أسئلة الطباعة، هناك عيب لهذا المشروع بحيث لا يقوم بالطباعة على الصفحات من اليمين لليسار، نعم يقوم بطباعة العربي بشكل سليم ولكنه يُلصِق السطر بالجانب الأيسر من الصفحات، قد يكون تعديل هذه الخاصية سهل جداً ولا يتجاوز سطر من الكود إذا رجعنا إلى تعليمات الفيجوال بيسك Documentation.

خلاصة الفصل السابع عشر

من اجل أن	قم بالتالي
التسهيل في عملية كتابة أكواد الطباعة	نكتب الكود التالي في أعلى منطقة الكود: Imports System.Drawing.Printing
لمراقبة حدث الطباعة	نستخدم الجملة AddHandler مع AddressOf ثم المعامل كما في الكود: AddHandler PrintDocument1.PrintPage, _ AddressOf Me.PrintGraphic
إنشاء الكائن PrintDocument في مشروعك	نستطيع إضافة الكائن PrintDocument من صندوق الأدوات، أو إضافته بطريقة برمجية كما في الكود التالي: Dim PrintDoc As New PrintDocument

طباعة الصور من مراقب أحداث الطباعة	نستخدم الطريقة Graphics.DrawImage لطباعة الصور كالتالي: ev.Graphics.DrawImage(Image.FromFile _ (TextBox1.Text), ev.Graphics.VisibleClipBounds)
طباعة نصوص من مراقب أحداث الطباعة	نستخدم الطريقة Graphics.DrawString لطباعة النصوص، مثال: ev.Graphics.DrawString(TextBox1.Text, _ New Font("Arial", 11, FontStyle.Regular), _ Brushes.Black, 120, 120)
استدعاء مراقب أحداث الطباعة	نستخدم الطريقة Print كالتالي: PrintDoc.Print()
طباعة نص يحتوي على أكثر من صفحة	إنشاء صفحة وهمية (صفحة افتراضية) تسمى PrintPage ثم إرسال النص إليها حتى تمتلئ ومنها نرسل الصفحة كاملة إلى الطابعة ويتم إعادة هذا الإجراء حتى تنتهي الصفحات التي نقوم بطباعتها، هناك طريقة أُخرى لطباعة النص متعدد الصفحات باستخدام الطريقة Graphics.MeasureString حيث تقوم هذه الطريقة بحساب عدد الأحرف اللازم طباعتها ونوع الخط وحجم الحروف ثم حساب مقاسات الصفحة ثم تنفيذ الطباعة حتى آخر صفحة.
فتح ملف نصي باستخدام الفئة FileStream ثم قراءة محتويات الملف	نقوم بتعريف متغير من النوع FileStream ثم نحدد مسار وطبيعة الملف، ثم نقوم بتحميل محتويات الملف النصي بطريقة انسيابية باستخدام الطريقة FileStream ثم نغلق مسار الانسياب Stream كما في المثال: Imports System.IO 'في بداية منطقة الكود لاستيراد الفئة Stream

إلى صندوق نص	... Dim MyFileStream As New FileStream(_ FilePath, FileMode.Open) RichTextBox1.LoadFile(MyFileStream, _ RichTextBoxStreamType.PlainText) MyFileStream.Close()
إظهار نوافذ الطباعة في برنامجك	نقوم بإضافة المكونات PrintDialog, PrintPreviewDialog، و PageSetupDialog إلى برنامجك ثم كتابة الأكواد الخاصة بهم.

الجزء الثالث: قواعد البيانات وبرمجة إنترنت (الويب)

في الجزء الثالث من هذا الكتاب سوف تتعلم كيف تتعامل مع البيانات المُخَزَنة في قواعد البيانات وفي مواقع إنترنت، في البداية نستعلم كيفية التعامل مع قواعد البيانات باستخدام تقنية ADO.NET وهي تقنية هامة للتعامل مع قواعد البيانات، وسوف نتعلم كيفية عرض البيانات وتعديلها والبحث فيها باستخدام بعض الأدوات وكذلك الكود، في الفيجوال بيسك ٢٠٠٨ تم تحديث العديد من الأساليب من أجل التعامل الأمثل مع قواعد البيانات، لن نتعامل مع قواعد البيانات على أنها سجلات من البيانات ولكن هناك العديد من التقنيات للتعامل مع البيانات بكل احترافية.

الفصل الثامن عشر: البداية مع تقنية ADO.NET

برمجة قواعد البيانات باستخدام تقنية ADO.NET

قواعد البيانات: هي مجموعة من البيانات المتجانسة والمنظمة المخزونة في ملف. نستطيع إنشاء ملفات قواعد البيانات باستخدام البرامج الخاصة بقواعد البيانات مثل مايكروسوفت أكسس

Microsoft Access أو إس كيو إل Microsoft SQL أو My SQL أو Oracle وغيرها من المنتجات التي تسهل علينا إنشاء ملفات قواعد البيانات، كذلك نستطيع تخزين ونقل البيانات باستخدام ملفات الـــ XML. قواعد البيانات هامة جداً للعديد من التطبيقات، فكل التطبيقات تقريباً تحتاج لقواعد البيانات، وكل المنشآت التجارية والعامية تحتاج لتطبيقات قواعد البيانات، فدفتر عناوين العملاء و تنظيم المخازن والفواتير وحسابات العملاء وإدارة الأصول وسجلات الموظفين كلها تعتمد على قواعد البيانات. نستطيع إنشاء قواعد بيانات جديدة باستخدام الفيجوال بيسك ٢٠٠٨ لكن الفيجوال بيسك ٢٠٠٨ صُمم لغرض التعامل مع قواعد الموجودة مسبقاً، لاستعراض البيانات والتعديل عليها.

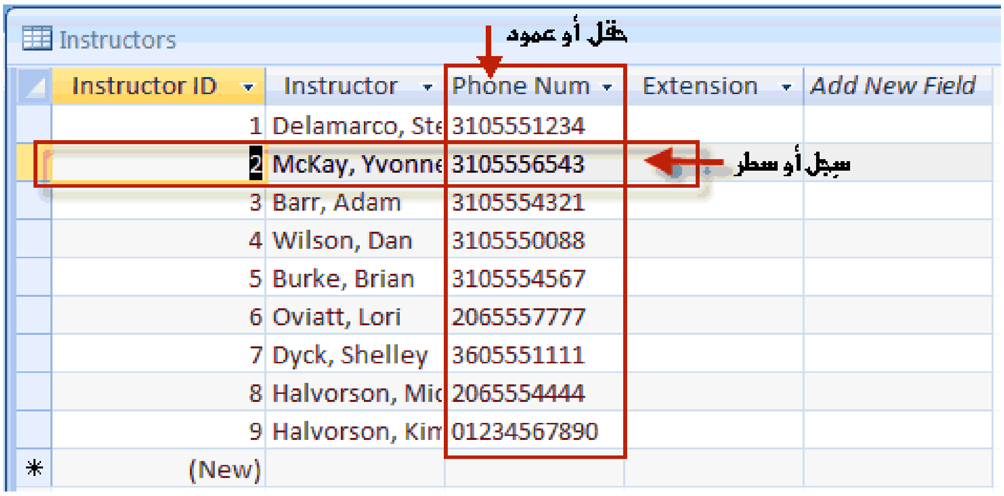
تطبيقات قواعد البيانات المصممة الفيجوال بيسك ٢٠٠٥ تعمل بكفاءة في نسخة الفيجوال بيسك ٢٠٠٨، فمعظم الأساليب المستخدمة في نسخة ٢٠٠٥ لا زالت هي نفسها في نسخة ٢٠٠٨، مع وجود تطويرات في نسخة ٢٠٠٨، وأهم هذه التطويرات هما Language-Integrated Query (LINQ) وكذلك ADO.NET Entity Framework ، لابد من استخدام التقنيتين لمن يريد احتراف التعامل مع قواعد البيانات.

LINQ تم تضمينها في نسخة ٢٠٠٨ وتسهل لنا تصميم أوامر للتعامل مع قواعد البينات بطريقة البرمجة كائنية التوجه التي تختصر لنا الوقت والجهد. وتم إصدار تقنية ADO.NET Entity Framework بعدها والتي أضافت كائنات وميزات جديدة لتسهيل التعامل مع قواعد البيانات.

مصطلحات قواعد البيانات

مع الوقت تتطور تقنيات قواعد البيانات وتظهر تقنيات مساعدة كثيرة، تعلم كل التقنيات الجديدة قد يصيب المبرمج بالإحباط فلابد من التعامل مع المسألة من منظور الحاجة فإذا احتجنا لتقنية معينة أو ميزة معينة فلنبحث عنها، الإطار العام لقواعد البيانات لم يتغير منذ ثمانينات القرن الماضي، هناك العديد من التقنيات نعم لكن الإطار العام لم يتغير، لمعرفة فكرة عمل قواعد البيانات لابد من قراءة موضوع لمحترف قواعد بيانات في موقع الفريق العربي للبرمجة، رقم الموضوع في المنتدى هو ٣٥٢٣٧، هذا الموضوع يضع بين يديك ماهية قواعد البيانات بغض النظر عن البرنامج المستخدم في التعامل معها، نعود لمسألة مصطلحات قواعد البيانات، الحقل Field (أو

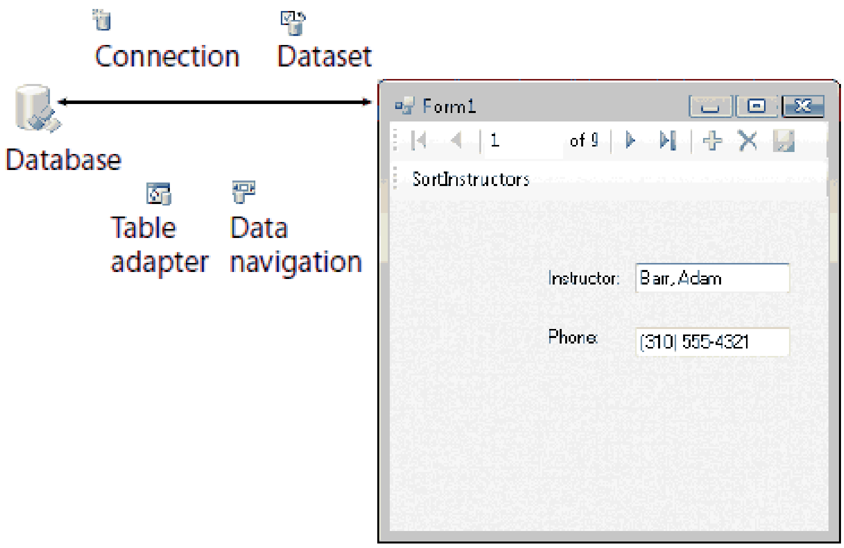
العمود) هو مجموعة من البيانات من نفس النوع المُخزنة بداخل قاعدة البيانات، هذه البيانات قد تكون أرقام هواتف العملاء أو عناوين شركاتهم أو ملاحظات. كل البيانات عن شركة واحدة تسمى سجل Record (سطر)، وعندما يتم إنشاء قواعد البيانات تتم عملية إدخال البيانات في الأعمدة والأسطر، تتقاطع الحقول مع السجلات أو الأعمدة مع الأسطر كما في الشكل:



Instructor ID	Instructor	Phone Num	Extension	Add New Field
1	Delamarco, Ste	3105551234		
2	McKay, Yvonne	3105556543		
3	Barr, Adam	3105554321		
4	Wilson, Dan	3105550088		
5	Burke, Brian	3105554567		
6	Oviatt, Lori	2065557777		
7	Dyck, Shelley	3605551111		
8	Halvorson, Mic	2065554444		
9	Halvorson, Kim	01234567890		
(New)				

قواعد البيانات العلائقية Relational Database هي قواعد البيانات التي تحتوي على أكثر من جدول تربط بينهم علاقات، فمثلاً جدول للعملاء وعناوينهم وجدول بالفواتير المستحقة عليهم. معظم قواعد البيانات التي سنتعامل معها بالفيجوال بيسك ٢٠٠٨ هو قواعد بيانات علائقية لأنها تحتوي على أكثر من جدول مرتبطين بعضهم ببعض.

تقنية الـــ ADO.NET تستخدم أكثر من مكون للتعامل مع قواعد البينات الشكل التالي يوضح المدخل الذي سنتعلمه في هذا الفصل للتعامل مع قواعد البيانات:



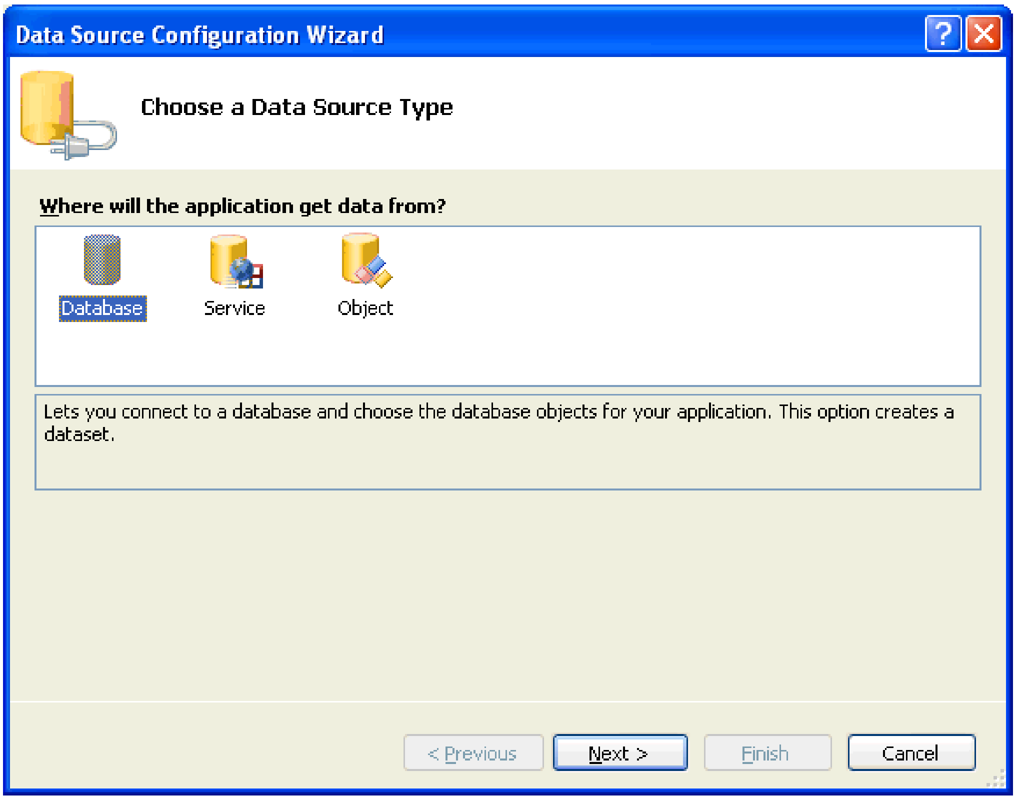
أولاً هناك مُوصِل Connection، والذي يحتوي على معلومات عن قاعدة البيانات ويحتوي على معلومات تستخدمُها بقية المكونات للربط مع قاعدة البيانات، إذا كانت قاعدة البيانات تحتوي على كلمة سر فهذا المُوصِل يحتوي على كلمة السر لقاعدة البيانات، بعد قاعدة البيانات هناك ما يسمى بـــ Dataset وهي عبارة عن (عارض) لما تحتويه الجداول أو الجدول بداخل قاعدة البيانات، بعد عملية الربط مع قاعدة البيانات يقوم محرك الربط مع قاعدة البيانات بإنشاء ملف XML ليسهل عملية الربط بين قاعدة البيانات وبين المكونات Dataset و table adapter و data navigator وكذلك تسهيل عملية التنقل بين البيانات أو تعديلها.

التعامل مع قاعدة بيانات مايكروسوفت أكسس Microsoft Access

في المرفق رقم ٥٥٠ ستجد قاعدة بيانات من نوع Microsoft Access باسم Students.mdb وهي عبارة عن قاعدة بيانات أكاديمية تعليمية عن مجموعة من الطلاب، يناسب الملف Microsoft Access نسخة ٢٠٠٢ و ٢٠٠٣ ويعمل على نسخة ٢٠٠٧، كما توجد نسخة من الملف لنسخة Access 2000 تحت اسم Students_2000format.mdb، البيانات بداخل قاعدة البيانات مفيدة لمن يريد مراجعة سجلات الطلاب سواء إدارة أو مدرسين، إذا لم يتوفر لديك Microsoft Access فلا تخف، فتوجد في الفيجوال بيسك ٢٠٠٨ وفي تقنية ADO.NET العديد من المكتبات التي تتعامل مع ملفات الأكسس وفهم محتوياتها بكل سهولة، نستفيد من ملف قاعدة البيانات في إنشاء أول مشروع في قواعد البيانات كالتالي:

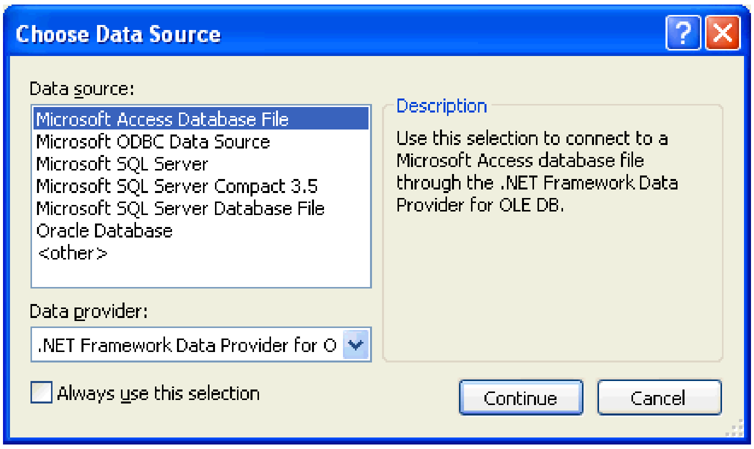
١- ننشئ مشروع جديد بواسطة الفيجوال بيسك ٢٠٠٨ ونسميه My ADO Form.

٢- من قائمة Data نختار Add New Data Source ستظهر لنا النافذة التالية:

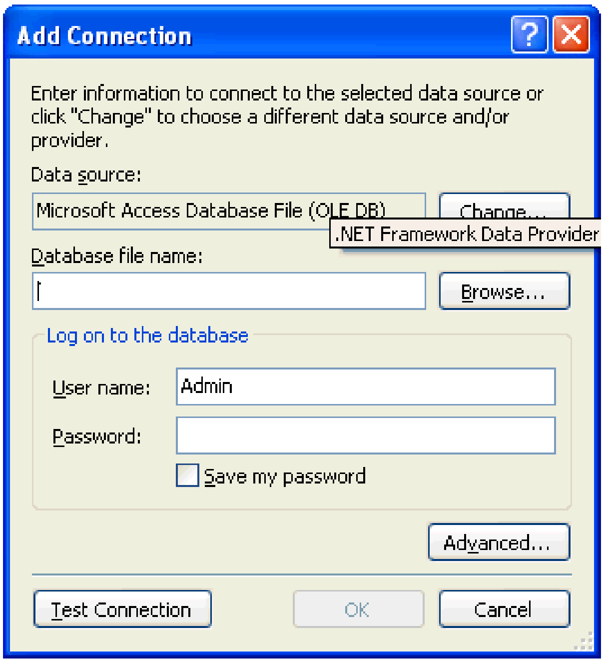


هذه النافذة تسمى Data Source Configuration Wizard وتسهل علينا عملية الربط مع قواعد البيانات، الخيار Database لفتح قاعدة بيانات محلية الخيار Service لفتح قاعدة بيانات على الوين والخيار الثالث لفتح كائن مُصمم سابقاً.

٣- نختر Database ثم Next سيظهر لنا نافذة جديدة (ستقوم هذه النافذة بتكوين لنا معلومات عن الاتصال مع قاعدة البيانات وذلك بأخذ كل المعلومات عن قاعدة البيانات وطريقة الربط ومسار قاعدة البيانات وكذلك اسم المستخدم وكلمة المرور لقاعدة البيانات) نختار منها نوع قاعدة البيانات بالضغط على الزر New Connection ستظهر هذه النافذة:

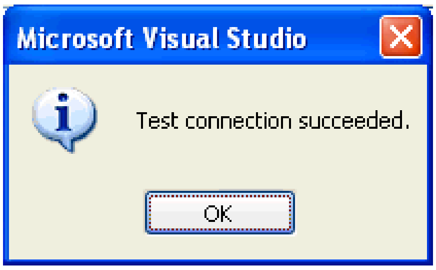


نختر منها Microsoft Access ثم Continue، تظهر لنا نافذة جديدة كالتالي:

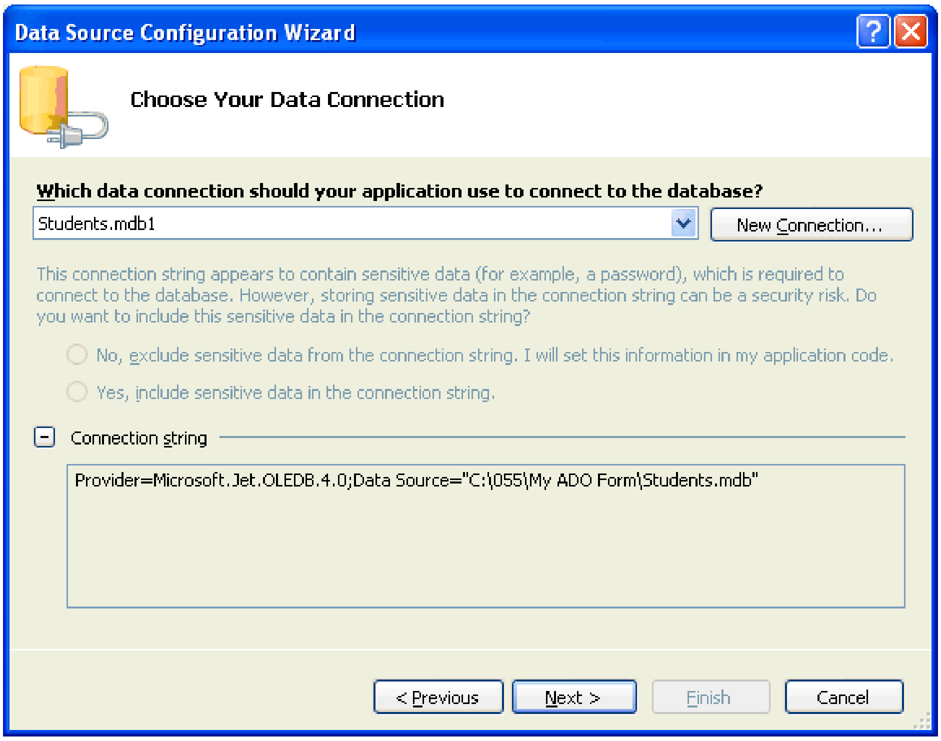


٤-من هذه النافذة نختر مسار قاعدة البيانات بالضغط على الزر Browse ثم اسم وكلمة المرور لقاعدة البيانات، طبعاً مسار قاعدة البيانات Student.mdb في المرفق رقم ٥٥٠، وبما أن قاعدة البيانات المذكورة لا تحتوي على اسم وكلمة سر فلا داعي لكتابتهما، وفي حالة كانت لديك قاعدة بيانات تحتوي على اسم وكلمة سر قم بكتابتهما ليتم إضافتهما إلى أوامر التواصل مع قاعدة البيانات. لنتأكد الآن من عملية التواصل مع قاعدة البيانات بالضغط على الزر تجربة التوصيل "Test Connection"، فإذا كان مسار قاعدة البيانات

صحيحاً وكان اسم المستخدم وكلمة السر صحيحاً (طبعاً في قاعدة البيانات هذه لا يوجد اسم وكلمة سر) ستظهر لك مثل هذه النافذة لتحبرك بنجاح العملية:



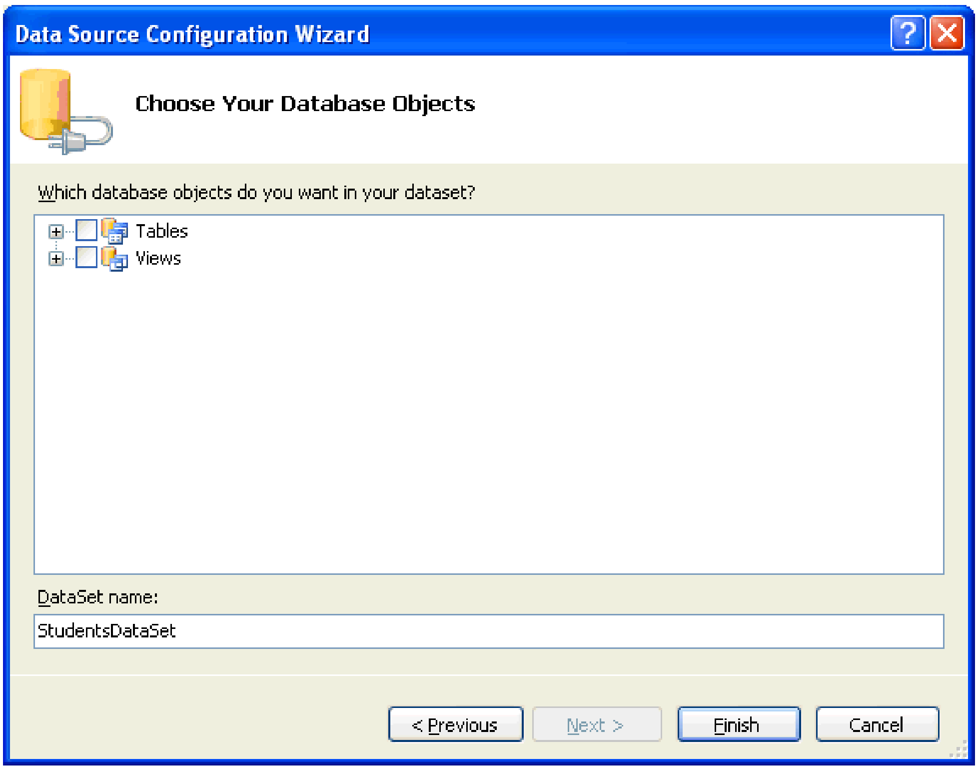
٥- نختر Ok سيقوم الفيجوال بيسك بإرجاعنا لمعالج الاتصال مع قاعدة البيانات مرة ثانية نضغط على الزر (+) لنرى كود الاتصال الذي كتبة المعالج حتى الآن كما في هذه الصورة:



كود الاتصال يحتوي على نوع الاتصال وذلك بتحديد الـــ Provider والمسمى Microsoft.Jet.OLEDB.4.0 ، (هناك نوعين من الـــ Provider نتعامل بواسطتهما مع قواعد البيانات وهما Microsoft OLE DB و Microsoft SQL Server وهناك العديد من الـــ Providers المقدمة من الشركات الأُخرى للتعامل مع بقية أنواع قواعد البيانات).

٦- نختار التالي Next ستظهر لنا صندوق حوار ليسأل فيما إذا كنا نرغب من الفيجوال بيسك بنقل قاعدة البيانات إلى مجلد البرنامج، في الوقت الحال لا نحتاج لذلك وعليه فنختار لا No، ستظهر لنا نافذة جديدة لتسألنا سؤال " Do you want to save the connection string to the application configuration file؟" هل تريد حفظ كود التوصيل مع قاعدة البيانات في ملف إعدادات البرنامج؟، الاختيار الافتراضي هو نعم Yes، ويفضل هذا الخيار ليقوم الفيجوال بيسك بخزن بيانات الاتصال في ملف إعدادات البرنامج والذي يوجد في مستكشف ملفات المشروع Solution Explorer من أجل إذا أردنا أن نغير مسار قاعدة البيانات نذهب إلى ملف إعدادات البرنامج في مستكشف المشروع Solution Explorer ثم تعديل المسار.

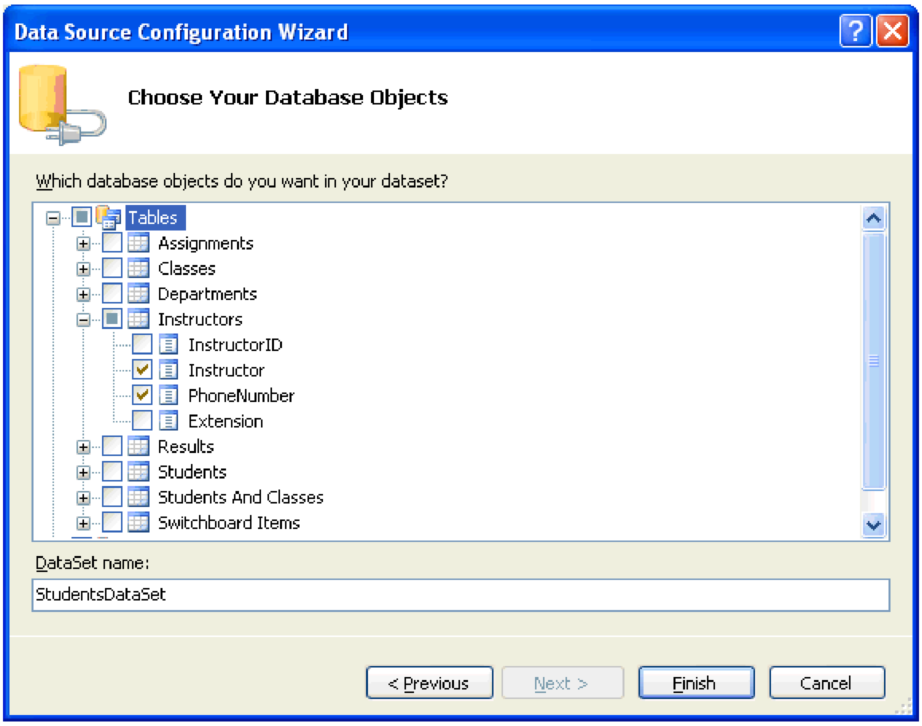
٧- نختار التالي Next ليقوم البرنامج بحفظ بيانات الاتصال، وسيفتح لنا نافذة جديدة لنقوم باختيار جزء من قاعدة البيانات (إذا كنا نحتاج فقط جزء منها) أو اختيار قاعدة البيانات كاملة:



وكما وضحنا إذا كانت قاعدة البيانات كبيرة جداً يمكننا اختيار الجزء الذي يخصنا فقط منها، البنود التي سنختارها من قاعدة البيانات ستسمى بداخل مشروعنا كائنات قاعدة البيانات database objects، وقد تحتوي هذه الكائنات على سجلات وجداول و...و غيرها، المصطلح الذي يستخدم لتسمية كل هذه الكائنات المُختارة هو dataset (مجموعة البيانات)، اسم مجموعة البيانات في الشاشة أعلاه StudentsDataSet نستطيع تغييره أو الإبقاء عليه.

ملاحظة: قمنا الآن بإنشاء مجموعة البيانات dataset للتعامل مع البيانات، إذا قمنا بحذف أو إضافة أو تعديل لمجموعة البيانات dataset فلن يقوم البرنامج بنقل هذه التعديلات لقاعدة البيانات الأصلية إلا إذا كتبنا كود برمجي للقيام بهذه العملية، لذلك تسمى هذه الطريقة من الاتصال بقاعدة البيانات بالطريقة المنفصلة أو **مصدر البيانات المنفصل**، أي أن هناك طبقة بين مصدر البيانات الأصلي وبين مجموعة البيانات عند المستخدم.

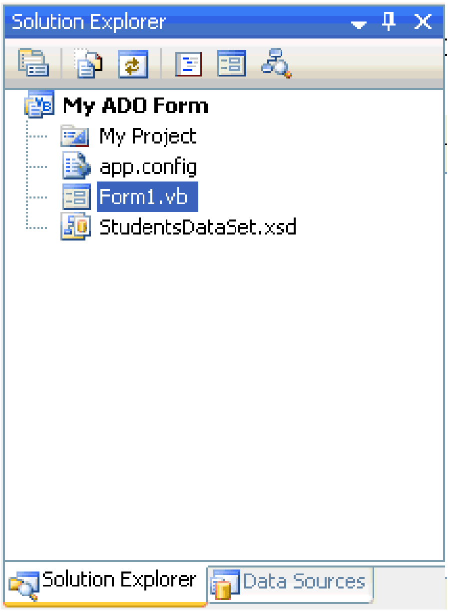
٨- نختار العلامة (+) أمام الجداول Tables ليتم عرض الجداول الموجودة بداخل قاعدة البيانات سنختار الجدول Instructors فقط من بين بقية الجداول، نختار العلامة (+) أما الجدول Instructors ونختار من الحقول الحقلين Instructor و PhoneNumber فقط، ستكون نافذة الاختيار كما في الصورة:



٩- نختر Finish لإكمال عملية التواصل مع قاعدة البيانات. الآن انتهى الفيجوال بيسك ٢٠٠٨ من التواصل مع قاعدة البيانات وتم تكوين Dataset (مجموعة بيانات) فقط بالبيانات المختارة.

١٠- نضغط على الزر Save All لحفظ الإجراءات التي قمنا بها.

١١- لنذهب الآن إلى مستكشف المشروع Solution Explorer الذي يكون كما في هذه الصورة:



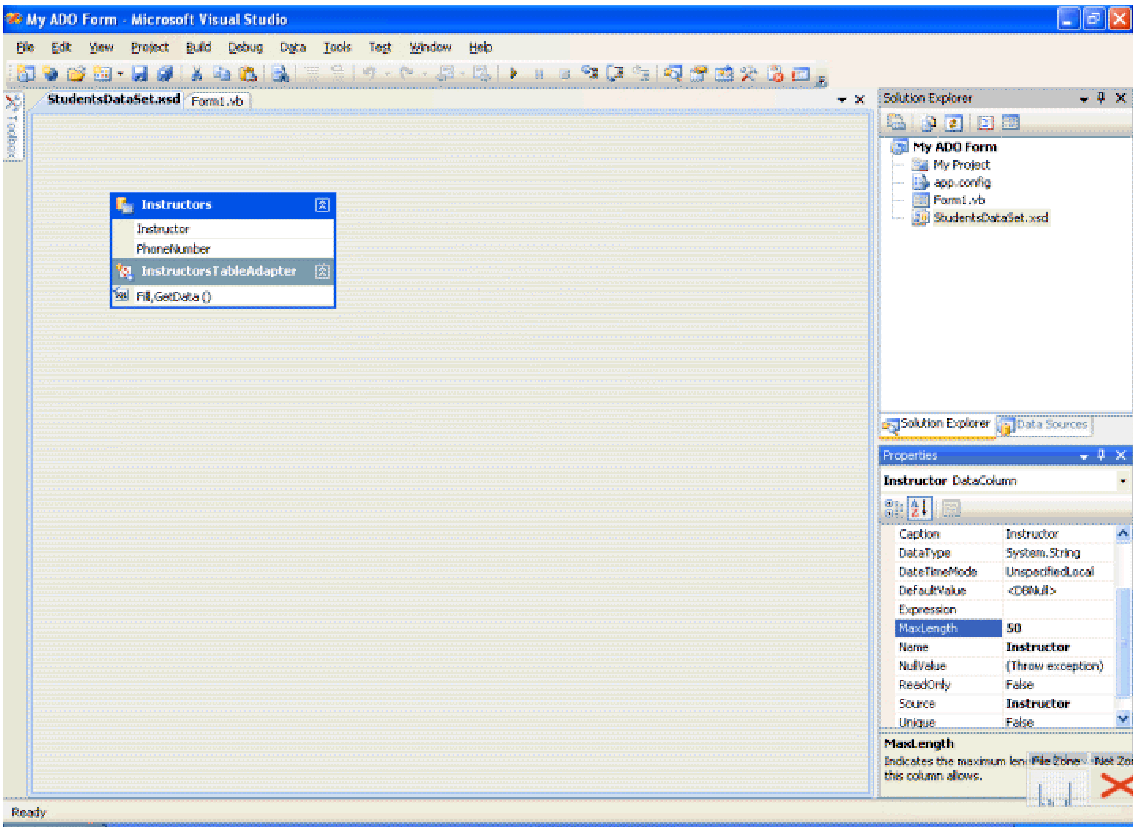
نرى في مستكشف المشروع ملف جديد وهو StudentsDataSet.xsd هذا هو الملف الذي تم تكوينه خلال عملية الاتصال مع قاعدة البيانات وهو من النوع XML، مجموعة البيانات النموذجية Typed datasets تحتوي على هذا النوع من الملفات لكن مجموعة البيانات الغير نموذجية un-typed datasets لا تحتوي على هذا النوع من الملفات، تعتبر مجموعة البيانات النموذجية أفضل من الغير نموذجية لأنها تفعل وتسهل عملية الإكمال التلقائي IntelliSense في محرر الكود وتقدم لك معلومات معينة عن الجداول والحقول التي تقوم بكتابتها في محرر الكود.

١٢- ننقر على الملف StudentsDataSet.xsd، سنرى عرض لما يحتويه هذا الملف وذلك بعرض سجلات قاعدة البيانات، يسمى هذا مصمم مجموعة البيانات

Dataset Designer هذا المصمم يحتوي على الكائنات التي تقوم بالتوصيل بين برنامجك وقاعدة البيانات، من خلال هذا المصمم تستطيع أن تكتب أوامر لقواعد البيانات والتعامل معها وتستطيع مراجعة وتعديل خصائص حقول قواعد البيانات.

١٣- قم بالنقر على الحقل Instructor ثم اضغط على F4 لننتقل للعمل على الخصائص Properties.

١٤- نختار الخاصية MaxLength من الخصائص، ستكون شاشتنا كما في هذه الصورة:

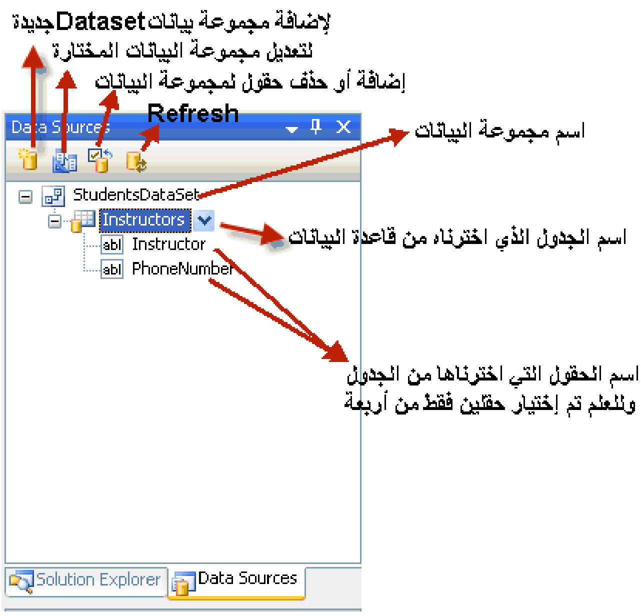


إذا لاحظنا الخاصية MaxLength وتُعني (عدد الخانات في هذا الحقل) سنرى الرقم ٥٠ فيها بحيث يسمح لنا البرنامج بخزن خمسين حرف أو رقم في هذا الحقل، طبعاً الرقم ٥٠ كافي ولا نحتاج لتعديله، ويمكننا في وقت الحاجة.

شاشة مصدر البيانات Data Sources Window

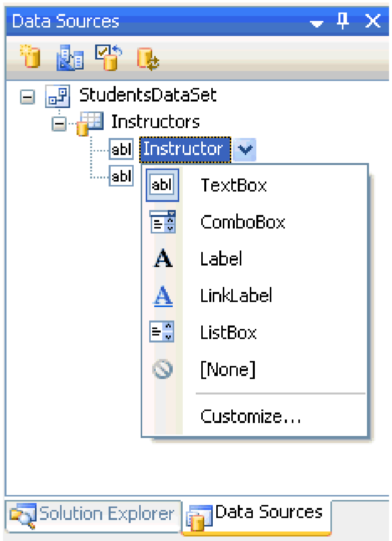
تعتبر شاشة مصادر البيانات ميزة رائعة في بيئة التطوير ٢٠٠٨، إنها تختصر الوقت علينا في التعامل مع قواعد البيانات، إنها تقوم بعرض البيانات الموجودة في مجموعة البيانات Dataset في تطبيقنا، بل وتساعدنا على ربط البيانات في مجموعة البيانات Dataset مع عناصر التحكم في برنامجنا فتربط البيانات مع الأزرار وصناديق النص والمؤقتات وغيرها، علينا أن نتذكر الآن بأن مجموعة البيانات ليست هي قاعدة البيانات وإنما هي عبارة عن وسيط بين برنامجنا وبين قاعدة البيانات وكل مجموعة من البيانات Dataset هي عبارة عن جزء من الجداول والحقول ولا تمثل كل قاعدة البيانات (حيث تعرض فقط الجداول والحقول التي اخترناها خلال مرحلة الربط مع قاعدة البيانات الموضحة في بداية هذا الفصل)، يتم عرض مجموعة البيانات Dataset بشكل شجري Tree في شاشة مصادر البيانات Data Source Window، كل فرع يحتوي على الكائن الذي اخترته خلال عملية التصميم، كل مرة نقوم بتصميم مجموعة بيانات Dataset تقوم شاشة مصادر البيانات بإضافة فرع في الشجرة لتسهل علينا عملية ربط مجموعة البيانات Dataset مع عناصر التحكم على النموذج (الفورم)، قد تتساءل: أين توجد شاشة مصادر البيانات Data Sources ? الجواب توجد مُدمَجة مع مستكشف المشروع Soluation Explorer أعلى يمين بيئة التطوير، إذا لم تكن ظاهرة إذهب إلى النموذج Form1 في مرحلة التصميم ثم من القائمة Data اختر Show Data Sources أو بالضغط على الاختصار Shift+Alt+D.

بعد فتح شاشة مصادر البيانات نضغط على (+) بجانب Instructors لنرى الحقلين اللذين قمنا بإختيارهما في مرحلة الربط مع قاعدة البيانات، وستكون شاشه مصادر البيانات كما في الشكل التالي مع بعض التوضيحات:



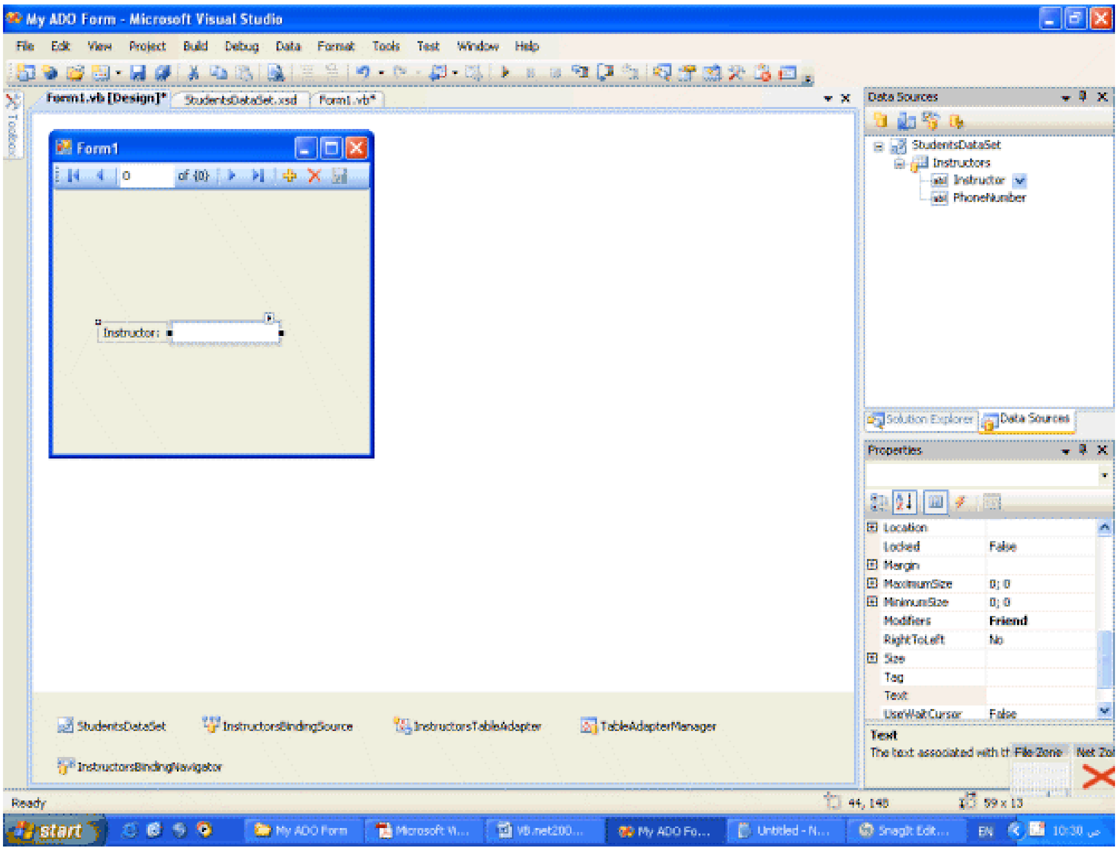
أسهل طريقة لإضافة المعلومات الموجودة ضمن مجموعة البيانات Dataset إلى تطبيقك هو استخدام الماوس في إضافة المكونات من شاشة مصادر البيانات Data Sources Window إلى سطح النموذج (الفورم)، في الفصل التاسع عشر سنتعلم كيفية إضافة جدول كامل إلى النموذج على شكل جدول DataGrid، لنقوم الآن بإضافة بعض العناصر من مجموعة البيانات Dataset إلى تطبيقنا:

١- نضغط على الحقل Instructor في مجموعة البيانات Dataset وسيفتح لنا قائمة بالمكونات التصميمية التي يمكن أن يتحول الحقل إليها كما في هذه الصورة:



لاحظ بأننا نستطيع عرض الحقل Instructor على النموذج بعدة طُرق وتقوم بيئة التطوير بعرض الطُرق أو المكونات الأكثر استخداماً، فيمكننا عرض النص في الحقل المذكور في صندوق نص أو ComboBox أو ليبل أو قائمة ListBox ويمكننا تخصيص المكونات التي تظهر بإضافة مكونات جديدة أو بإلغاء مكونات بالضغط على Customize. طبعاً الأنسب لنا في مثل هذه الحالة هو صندوق النص لعرض المحتويات النصية، تَذَكر بأنك تستطيع اختيار أياً من المكونات.

٢-فلنختر الآن صندوق النص ونقوم بإضافته (بواسطة السحب بالماوس) إلى وسط النموذج كما في الصورة:



شاهد التغييرات التي حدثت على النموذج بإضافة المكونات الجديدة بالأسفل وكذلك بعض الكائنات على النموذج والتي تسهل علينا عملية الربط مع الحقل Instructor كل اهذا بدون تدخل وتعب منا، لقد أضاف الفيجوال بيسك صندوق نصي ليقوم بعرض الحقل Instructor على النموذج وأضاف ليبل بنفس اسم الحقل Instructor وأضاف

مستعرض في أعلى النموذج ثم أضاف خمسة مكونات في الأسفل كنا سنستغرق يومَين كاملين لإضافتهم وكتابة أكوادهم من هذه المكونات:

StudentsDataSet مجموعة البيانات Dataset التي صممناها سابقاً والتي تعرض بعض الجداول وبعض الحقول من قاعدة البيانات Students.mdb.

InstructorsBindingSource عبارة عن مكون وسيط يقوم بالوصل بين الجدول Instructor والكائنات (صندوق النص أو غيره) التي تمثله على النموذج.

InstructorsTableAdapter عبارة عن مكون وسيط يقوم بالربط بين مجموعة البيانات StudentsDataSet وقاعدة البيانات Student.

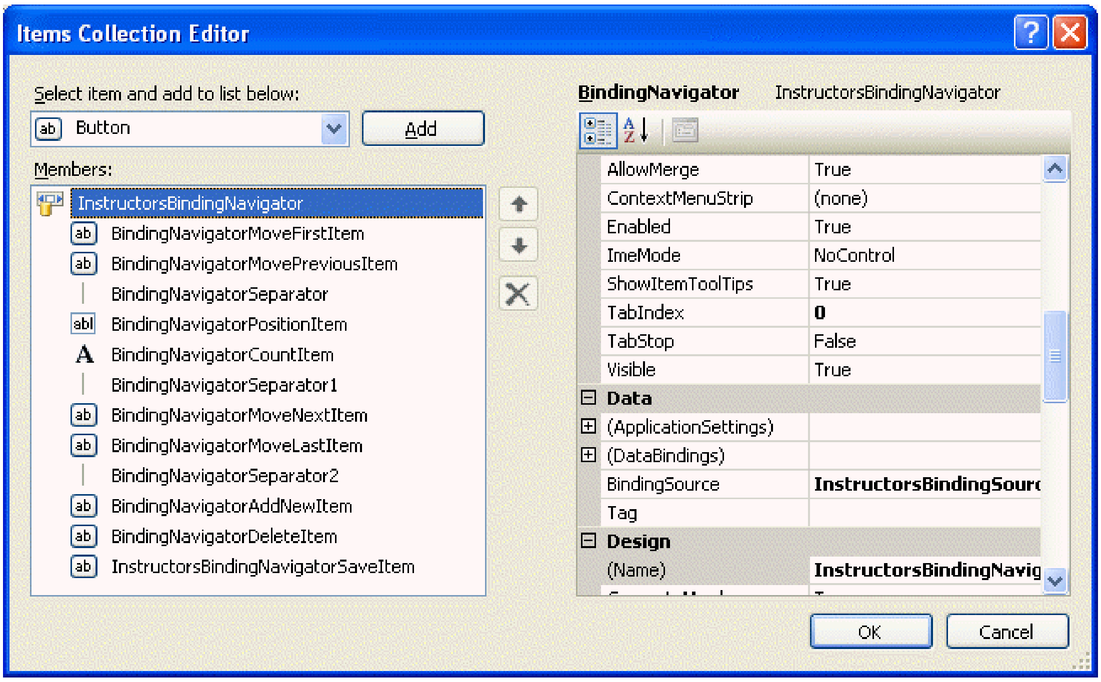
InstructorsBindingNavigator عبارة عن ومكون وسيط يسهل لنا عملية استعراض البيانات على النموذج بشكل يمكننا الانتقال من السجل الحالي إلى الذي يليه أو إلى السجل السابق ويمكننا هذا المستعرض من حذف أو إضافة سجل جديد لقاعدة البيانات.

سيلاحظ من له خبره بالفيجوال بيسك ٢٠٠٥ وجود نفس المكونات للتعامل مع قواعد البيانات، وهناك بعض الفروق مع فيجوال بيسك ٢٠٠٣.

٣-نقوم بتنفيذ البرنامج ليظهر لنا هذا البرنامج (مع التوضيحات):



نلاحظ أن المستعرض InstructorsBindingNavigator يسهل لنا الكثير في التنقل في قواعد البيانات، تستطيع حذف أو تعديل أحد أزرار المستعرض من خلال نافذة التصميم، فإذا أردنا عدم السماح للمستخدم بحذف السجلات بسهولة نحذف الزر الذي يقوم بالحذف خلال مرحلة التصميم، أو نستطيع اختيار المستعرض من نافذة الخصائص من الذهاب إلى الخاصية Items ونضغط على الزر ... ستظهر لنا هذه النافذة:



من هذه النافذة نستطيع عدم تفعيل المكون الذي لا نحتاج إليه كما نستطيع تعديل الخصائص للمكونات بالشكل الذي يناسبنا.

٤- نقوم الآن بتنفيذ البرنامج مرة ثانية وإضافة وإلغاء سجلات والذهاب إلى آخر سجل وإلى أول سجل، وكما وضحنا من قبل سيقوم البرنامج بحذف السجل أو بإضافته أو تعديله ولكن في مجموعة البيانات Dataset فقط، ولن ينقل هذه التعديلات إلى قاعدة البيانات الرئيسية Students.mdb، وللتأكد من ذلك نقوم بإعادة تشغيل البرنامج مرة أُخرى لنشاهد السجلات كما هي بدون تعديل، وكما وضَحنا من قبل نستطيع جعل البرنامج يحفظ التعديلات في قاعدة البيانات الرئيسية بواسطة الكود.

٥- إذا قمنا بإعادة تشغيل البرنامج مرة ثانية سنلاحظ عدد السجلات تسعة فقط، ومن هذا نستنتج بأن التعديلات التي تتم خلال عملية تشغيل البرنامج تحفظ في الذاكرة المؤقتة ولا

تحفظ في قاعدة البيانات، نقوم الآن بإغلاق التطبيق. مع ملاحظة أن نسخة من التطبيق في المرفق رقم ٠٥٥.

حتى الآن قمنا بتصميم تطبيق يتعامل مع قواعد البيانات بدون كتابة كود واحد، اخترنا جدول واحد من قاعدة البيانات ثم اخترنا حقلين فقط ثم اخترنا أحدهما للتعامل معه، بالطبع تستطيع اختيار أكثر من حقل متى ما أردت.

استخدام المكونات في استعراض قواعد البيانات

استخدمنا في المثال السابق طريقة لعرض المعلومات من قواعد البيانات، هذه الطريقة تتمثل بإدراج كائنات مجموعة البيانات Dataset من شاشة مصادر البيانات مباشرة إلى النموذج حيث نقوم باختيار الحقل ثم اختيار الطريقة المثلى التي يظهر بها الحقل (صندوق نص أو كائن آخر) على النموذج، لكن هل نستطيع إضافة المكونات إلى النموذج ثم ربطها مع قواعد البيانات بدلاً من استخدام المكونات المقترحة من نافذة مصادر البيانات، بعبارة أُخرى هل نستطيع إضافة أدوات ومكونات من صندوق الأدوات ثم ربط هذه المكونات بقواعد البيانات؟ نعم نستطيع ولنأخذ مثالاً على ذلك:

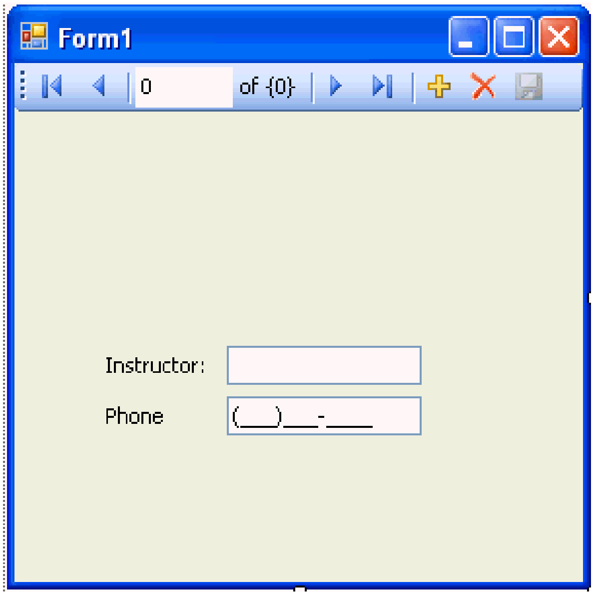
١- لنأخذ المثال السابق الذي قمنا بالتطبيق عليه والموجود بالمرفق رقم ٠٥٥.

٢- نفتح النموذج (الفورم) ونضيف الأداة MaskedTextBox من نافذة الأدوات، نضيف الأداة تحت الليبل وصندوق النص الموجودين مسبقاً، تعرفنا على الأداة MaskedTextBox في الفصل السادس من هذا الكتاب بحيث أنها نفس صندوق النص لكنها مناسبة أكثر لتنظيم المُدخلات في صندوق النص، بحيث تسمح للإدخال بطريقة معينة بحسب الطلب، حيث تسمح لنا الخاصية Mask بتعديل صيغة المدخلات، فبإمكان المدخلات أن تكون أرقام هواتف أو رقم عشري أو عملة أو غيرها، سوف نستخدم هذه الأداة لعرض أرقام الهاتف بشكل مميز، ففي قاعدة البيانات توجد أرقام هواتف الطلاب بدون مسافات أو أقواس لتسهيل عملية قراءة أرقام الهواتف لكننا سنضيف المسافات والأقواس في الأداة MaskedTextBox، **ملاحظة:** أرقام الهواتف المخزونة في قاعدة البيانات أرقام ليست لهواتف حقيقية، وإنما هي لهواتف مفترضة في أمريكا الشمالية وعليه

فطريقة عرض الأرقان في صندوق النص ستكون تبعاً للطريقة التي تعرض فيها أرقام الهواتف في أمريكا الشمالية.

٣-في مربع النص MaskedTextBox ننقر فوق السهم الصغير أعلى يمين مربع النص، نختار Set Mask ستظهر لنا نافذة تُظهِر لنا العديد من الإعدادات المقترحة والأكثر استخداماً، نختار منها Phone Number. (طبعاً نستطيع إعدادات خاصة حسب طبيعة المدخلات أو لأرقام الهواتف في بلداننا في الخانة الأخيرة Custom)، كما وضحنا سيضيف الإعدادات الخاصة بأمريكا الشمالية.

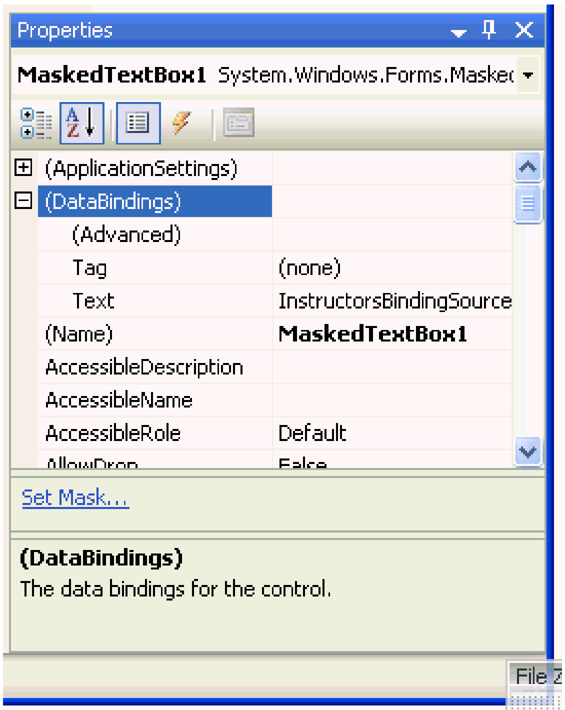
٤-نضيف ليبل وفي خاصية Text نكتب Phone، سيكون النموذج كما في هذه الصورة:



الآن سنقوم بربط الحقل PhoneNumber الموجودة في مجموعة البيانات StudentsDataSet بمربع النص MaskedTextBox1، وهذه العملية سهلة جداً في فيجوال بيسك ٢٠٠٥ و ٢٠٠٨، حيث نذهب إلى نافذة مصادر البيانات ونختار الحقل PhoneNumber ونسحبه بالماوس إلى داخل النموذج إلى داخل مربع النص.

٥-نستعرض نافذة مصادر البيانات، وسحب الحقل PhoneNumber إلى داخل صندوق النص MaskedTextBox1. وعند إضافة حقل إلى كائن على النموذج بهذه الطريقة يقوم الفيجوال بيسك بتعديل الخاصية DataBinding التابعة للكائن ففي صندوق النص MaskedTextBox1 قام الفيجوال بيسك بتعديل الخاصية DataBinding ليقوم بربط

صندوق النص بمجموعة البيانات وبالحقل PhoneNumber، انظر نافذة الخصائص لصندوق النص والخاصية DataBinding:



٦-نقوم الآن بتنفيذ البرنامج، ثم قم بالتنقل بين السجلات ستلاحظ أن البرنامج ينتقل بين السجلات في الحقلين في نفس الوقت.

٧-المثال في المرفق رقم ٠٥٦.

قبل أن ننتقل للفصل التاسع عشر لنستعرض قواعد البيانات على شكل جدول Grid، سنتعلم بعض التقنيات في التعامل مع قواعد البيانات بواسطة أوامر الـــ SQL وكذلك LINQ، من أجل فلترة البيانات.

خطوة للأمام: أوامر الـــ SQL و LINQ لفلترة البيانات

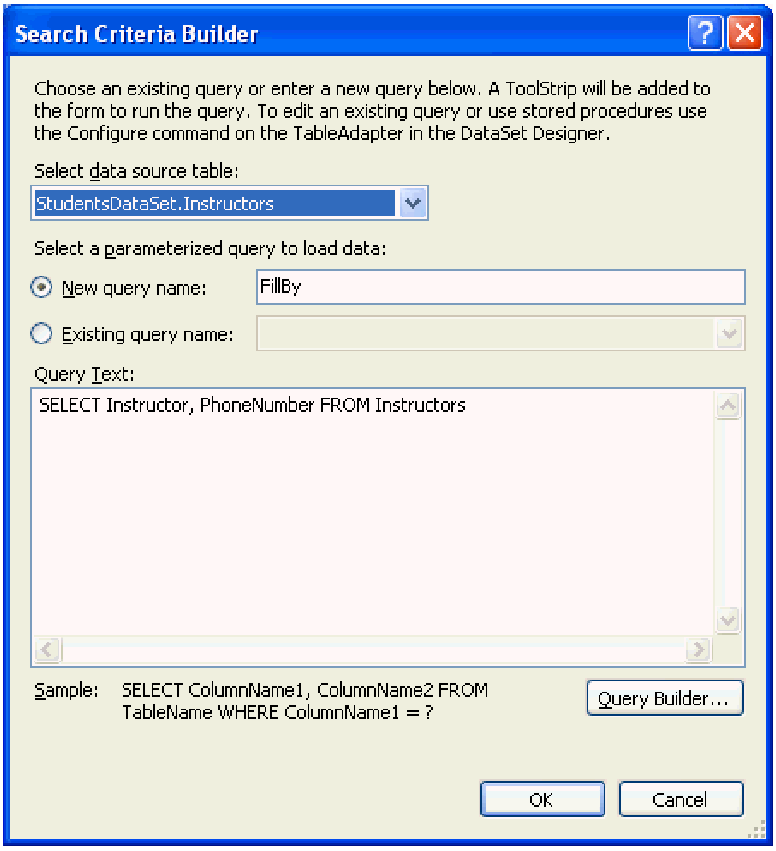
تعلمنا خلال هذا الفصل كيف نقوم بربط قاعدة بيانات بالكائنات على النموذج وكيف نقوم باستعراض محتويات قاعدة البيانات، عن طريق مجموعة البيانات StudentsDataSet، في أكثر الأحيان سنحتاج لاستخلاص بيانات معينة من قواعد البيانات، ففي قاعدة البيانات التي تحتوي على أكثر من مائة ألف سجل سنحتاج لاستخدام أوامر الـــ SQL لفلترة البيانات، فإذا كان لدينا

قاعدة بيانات تحتوي على بيانات العملاء عناوينهم وأرقام الهواتف لأكثر من خمسين ألف عميل وأردنا فقط العملاء المتواجدين في العاصمة أو المتواجدين في المدينة الساحلية الفلانية، ماذا إذا أردنا قائمة بالعملاء الذين تبدأ أرقام هواتفهم بالرقم ٨ (لأن هذا الرقم يتبع الخدمات المجانية)، ماذا وماذا، كل هذه الإجابات ستجدها مع أوامر الـــ SQL، أشهر أوامر الـــ SQL هو الأمر Select الشهير، باستخدام مجموعة من الأوامر يقوم المبرمج بكتابة الكود البرمجي المناسب للبحث عن معلومة معينة في قاعدة البيانات.

في نسخ فيجوال بيسك الأقدم من ٢٠٠٨ كان هناك دعم لأوامر الـــ SQL نفس الميزة موجودة في فيجوال بيسك ٢٠٠٨ مع إضافة جديدة وهي LINQ والتي تسمح للمبرمج بكتابة أوامر على هيئة SQL ضمن أكواد الفيجوال بيسك، وبالرغم من سهولة LINQ لكنها عصيه على الاحتراف ولن تحترفها حتى تحترف SQL، في المثال التالي سنستخدم ما يسمى بـــ Query Builder أو (بَنَاء الأوامر (اعتقد أن الترجمة غير مناسبة)) وتستخدم هذه الأداة لكتابة أوامر للتعامل مع قواعد البيانات، هذه الأداة تستخدم أوامر مشابهه للـــ SQL. سنستخدم Query Builder مع نفس المثال السابق وستقوم هذه الأداة بترتيب البيانات بشكل ألفبائي:

١- على النموذج ننقر على صندوق النص InstructorTextBox.

٢-من قائمة Data نختار Add Query سيقوم الفيجوال بيسك بفتح هذه النافذة:

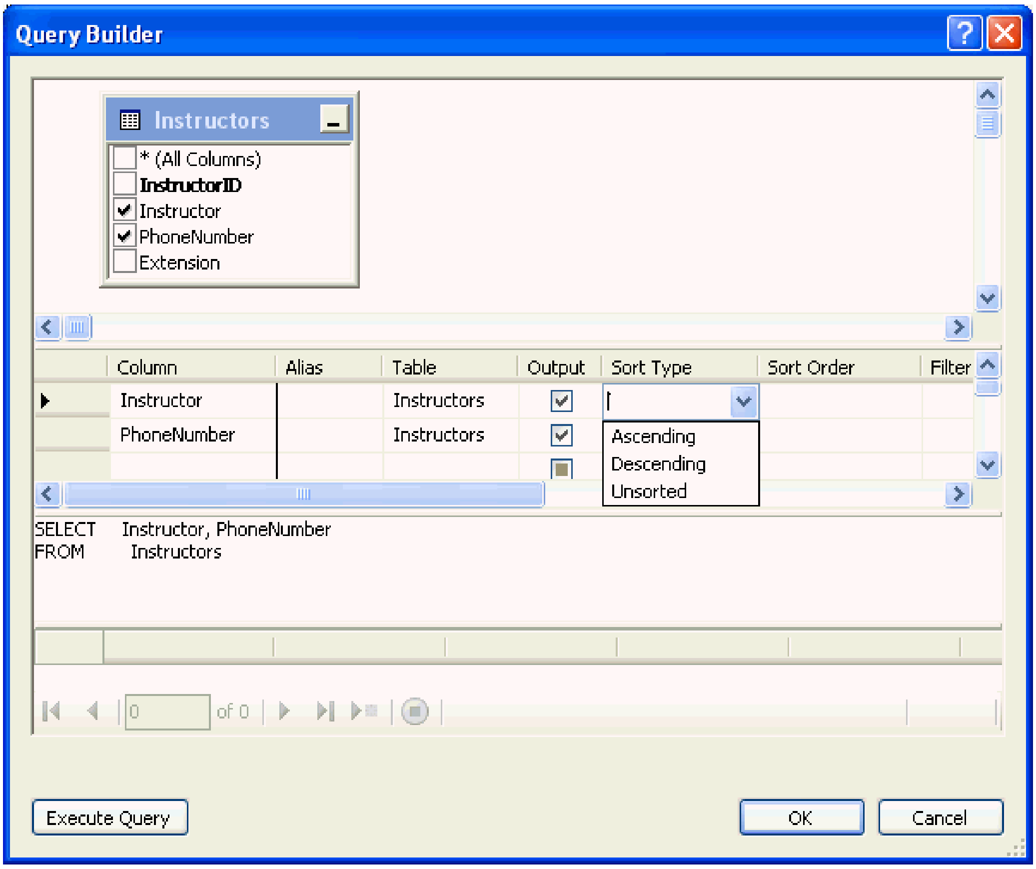


فالأداة Query Builder هي عبارة عن بَناء أوامر مرئي على هيئة SQL، انظر الكود المكتوب في Query Text وهو الكود الذي يقوم البرنامج بتطبيقه حالياً، إذا كان لدينا أكثر من جدول فسنجد الكود مكتوب كاملاً هنا.

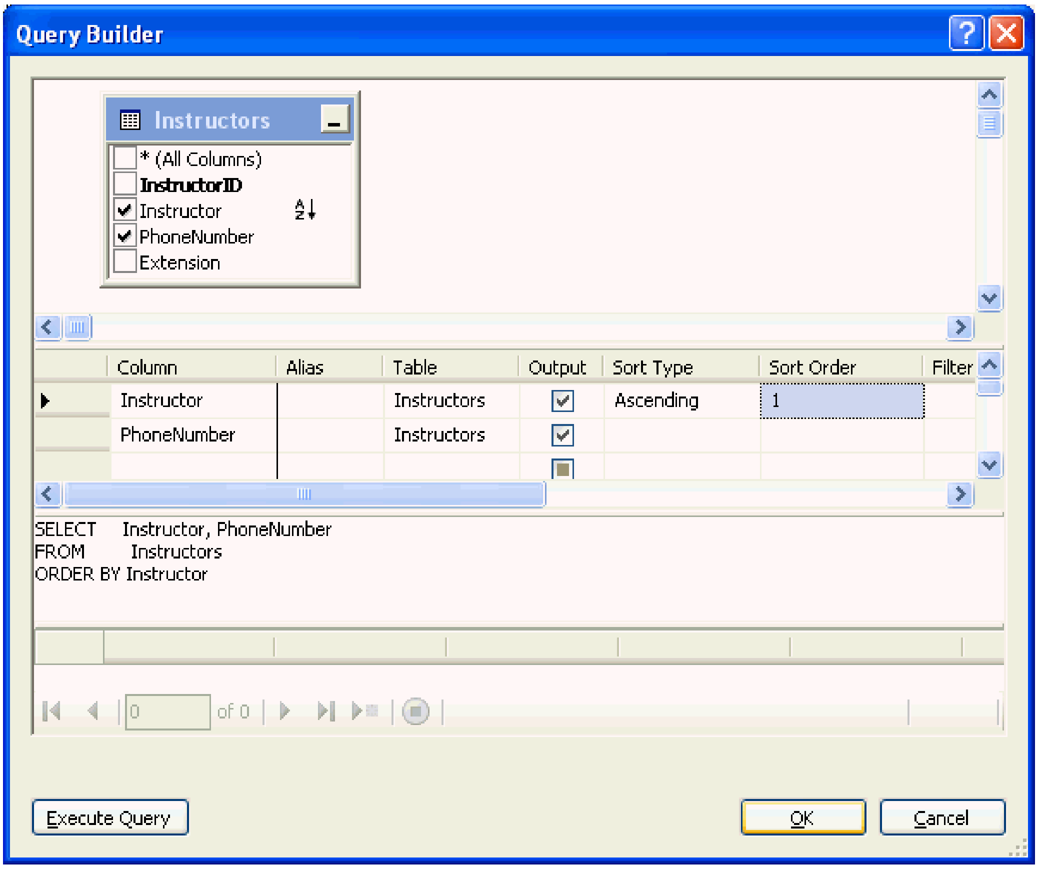
٣-قم بكتابة SortInstructors مقابل New Query Name، هذا اسم الأمر ونستطيع الاستفادة من الاسم في مرحلة الكود بإسناد الأمر إلى حدث يتبع كائن معين.

٤-الآن ننقر على الزر Query Builder ليفتح لنا نافذة جديدة، في النافذة الجديدة تستطيع كتابة أوامر الـــ SQL أو اختيارها بشكل مرئي في النافذة وسيقوم الفيجوال بيسك بكتابة الأوامر.

٥-لاحظ أن كل سطر في الجدول يمثل حقل من حقول قاعدة البيانات:



نذهب الآن للعمود Sort Type ونضغط مقابل Instructor ليفتح لنا قائمة بالخيارات الممكنة لترتيب الحقل، الخيارات هي Ascending تصاعدي و Descending تنازلي و Unsorted بدون ترتيب، نختار الترتيب التصاعدي Ascending، سنلاحظ بأن الفيجوال بيسك قام بإضافة الأمر ORDER BY Instructor إلى خانة الأوامر بالأسفل لتصبح كما في هذه الصورة:



نضغط Ok.

٦-ثم اضغط Ok لإغلاق النافذة الأولى، الأمر الذي استخدمناه الآن لا يقوم بفلترة البيانات وإنما يقوم بترتيب البيانات، سيقوم الفيجوال بيسك بإضافة زر جديد على النموذج باسم SortInstructors_ وسيكون النموذج كما في هذا الشكل:



كما سيضيف الفيجوال بيسك مكون ToolStrip جديد على النموذج وفي خانة المكونات.

٧- نقوم بتنفيذ البرنامج، ثم الضغط على الزر SortInstructors_ ليقوم البرنامج بترتيب السجلات بشكل ألفبائي حيث سيبدأ بالسجل Barr, Adam لإنه الأول في حالة الترتيب الألفبائي.

٨- نسخة من التطبيق في المرفق ٠٥٧، الآن قمنا بتطبيق أمر واحد من أوامر الـــ SQL وهو ترتيب السجلات بشكل ألفبائي تصاعدياً، هناك العديد من أوامر الـــ SQL التي تفيدنا في عملية فلترة قواعد البيانات، تستطيع البحث أو الاستفادة من الكتب المتخصصة، هنا بدأنا في أول الطريق والبقية عليك.

خلاصة الفصل الثامن عشر

من اجل أن	قم بالتالي
تتواصل مع قاعدة البيانات	من قائمة Data نختار Add New Data Source ثم قم بتحديد قاعدة البيانات للتواصل معها.
إنشاء مجموعة البيانات dataset	من النافذة Data Source Configuration Wizard نستطيع أن نختار اسم لمجموعة البيانات dataset ثم اختيار الجداول المراد اختيارها من

	قاعدة البيانات ثم اختيار الحقول من الجداول (الحقول والجداول المُختارة تسمى مجموعة البيانات dataset)، ليس بالضرورة أن نختار كل الجداول وكل الحقول وإنما ما نحتاجه فقط سيسمى مجموعة البيانات .dataset
اختيار حقول معينة لعرضها على النموذج	بعد التوصيل مع قاعدة البيانات واختيار الـــ dataset نذهب إلى نافذة مصادر البيانات بجانب مستكشف المشروع ثم نختار الحقول الهدف ونضيفها إلى النموذج.
إضافة مستعرض للانتقال بين السجلات	في فيجوال بيسك ٢٠٠٥ و ٢٠٠٨ تتم عملية إضافة المستعرض للنموذج أوتوماتيكياً مباشرة بعد إضافة كائن يرتبط مع حقل من حقول البيانات من النافذة مصادر البيانات Data Sources، ونستطيع التعديل على أزرار المستعرض بالتعديل على الخاصية Items في الكائن .InstructorBindingNavigator
عرض البيانات على النموذج بهيئات معينة	نستخدم الكائن MaskedTextBox لعرض البيانات عليه، ونعدل في الخاصية Mask للكائن ليعرض البيانات على الهيئة التي نريد.
فلترة أو ترتيب البيانات في مجموعة البيانات dataset	نستخدم أوامر الـــ SQL لفلترة البيانات، ونستخدم الأداة Query Builder لتساعدنا على كتابة الأوامر، بعد التعامل مع الأداة Query Builder بشكل احترافي تستطيع الانتقال إلى LINQ.

الفصل التاسع عشر: استعراض البيانات على شكل جدول بواسطة الكائن DataGridView

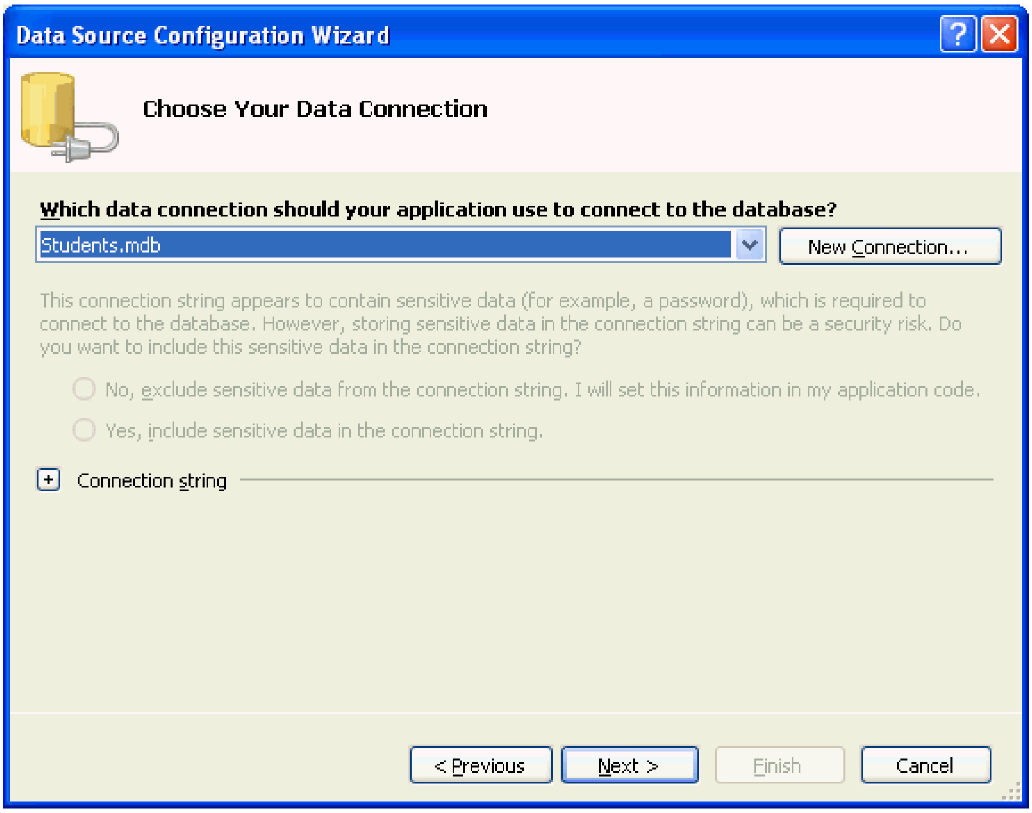
استخدام كائن الجدول DataGridView لعرض سجلات قاعدة البيانات

الكائن DataGridView هو عبارة عن كائن تتم إضافته إلى النموذج ليقوم بعرض جدول كامل من قاعدة البيانات، قد يحتوي هذا الجدول على أكثر من مائة سجل، طريقة العرض هي طريقة الجدول بحيث يعرض الحقول في أعمدة والسجلات في أسطر، أقرب مثال لهذا الجدول هو ملفات الأكسل أو الأكسس فهي تتشابه مع هذا الكائن إلى حد كبير، الكائن DataGridView في فيجوال بيسك ٢٠٠٨ و ٢٠٠٥ يختلف عن نفس الكائن في النسخ السابقة من الفيجوال بيسك، ذلك بأن الفرق هو إمكانية عرض البيانات على الـــ DataGridView مباشرة باستخدام مجموعة البيانات Dataset، يتم ربط البيانات مع الكائن بواسطة الخاصية BindingSource، طبعاً هذا بعد عملية ربط البيانات بواسطة النافذة Data Source Configuration Wizard واستخدام نافذة مصادر البيانات التي تعملنا كيف نستخدمهم في الفصل الثامن عشر، بعد عملية الربط مع البيانات يقوم الفيجوال بيسك بتعبئة الجدول DataGridView مباشرة بعد تحميل النموذج.

مثال على التعامل مع الكائن DataGridView

١- نقوم بإنشاء تطبيق جديد باسم My DataGridView Sample.

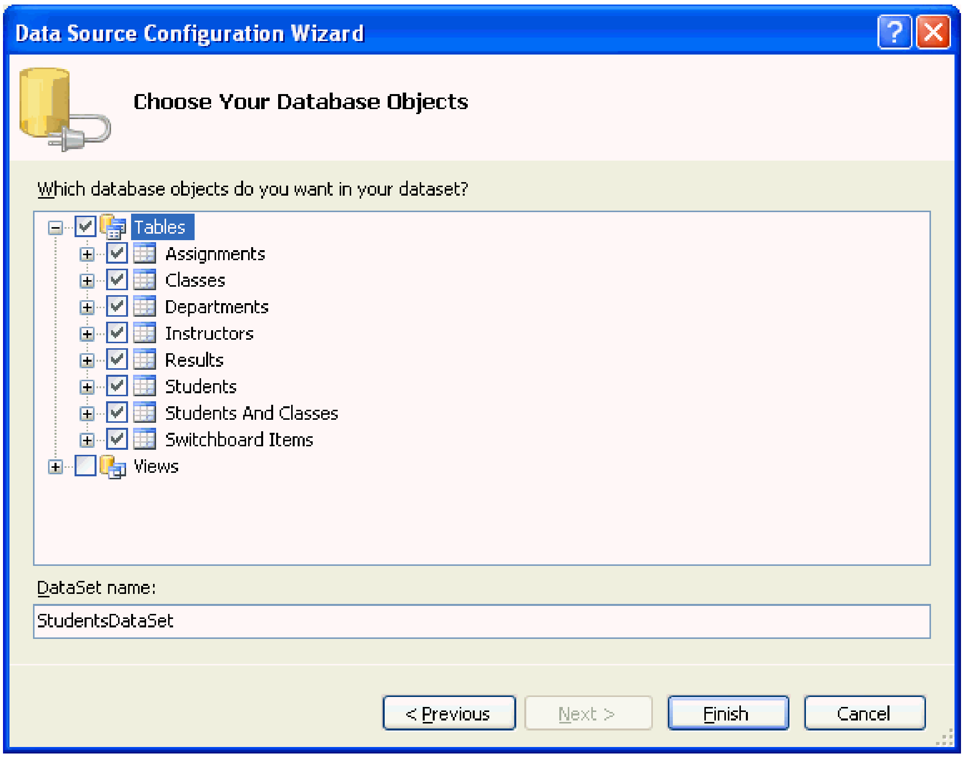
٢- من قائمة Data نختار Add New Data Source، (قد سبق لنا الربط مع قاعدة البيانات في الفصل الثامن عشر لكننا هنا سنختار حقول أكثر) نختار Database، طبعاً سيسمح لنا باختيار قاعدة البيانات، أما إذا قد قمت بتطبيق المثال في الفصل الثامن عشر فسيقوم الفيجوال بيسك باختيار قاعدة البيانات Student.mdb بشكل أوتوماتيكي كما في هذه الصورة:



إذا لم يقم الفيجوال بيسك باختيار قاعدة البيانات التي تريدها، قم بالضغط على New Connection ثم قم باختيار قاعدة البيانات Students.mdb الموجودة في المرفق رقم ٠٠٥٥، واتبع تفاصيل الربط مع قاعدة البيانات المتبعة في الفصل الثامن عشر، قم بالضغط على التالي ثم التالي.

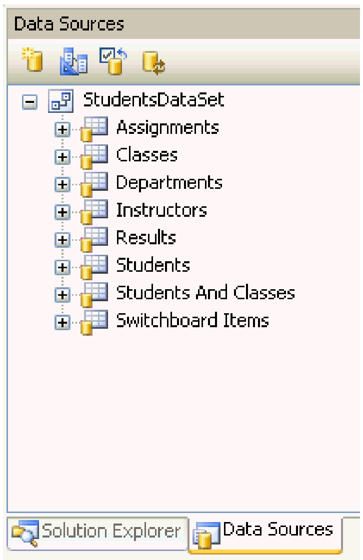
٣- الآن لديك خيارات لاختيار الجداول والحقول لإضافتها لمجموعة البيانات dataset، تستطيع اختيار كل الجداول والحقول أو اختيار ما تحتاجه فقط.

٤- نضغط على (+) بجانب الجداولTables سنجد أن هناك سبعة جداول والثامن يسمى Switchboard Items، نضغط على المربع الفارغ بجانب Tables لنقوم باختيار كل الجداول في قاعدة البيانات لأن هدفنا في هذا المثال استعراض قوة الكائن DataGridView في عرض البيانات، على كل ستكون نافذة الاختيار كما في هذه الصورة:

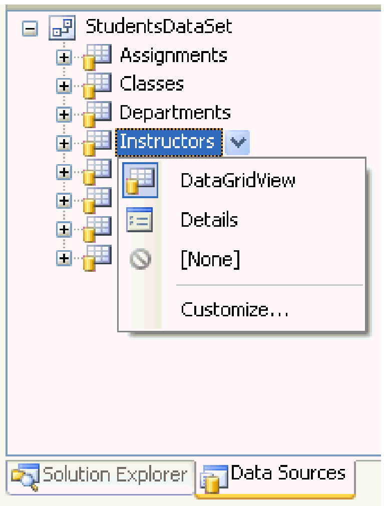


نضغط Finish لإكمال عملية الاختيار. سيقوم الفيجوال بيسك بإنشاء مجموعة البيانات StudentsDataSet.

٥- نفتح نافذة مصادر البيانات Data Sources بجانب مستكشف المشروع، ستحتوي نافذة مصادر البيانات على كل الجداول الموجودة في مجموعة البيانات StudentsDataSet كما في الشكل التالي:



٦-في الفصل الثامن عشر تعلمنا كيفية إضافة حقل واحد من جدول من قاعدة البيانات إلى كائن معين على النموذج أما الآن سنتعلم كيفية إضافة جدول كامل إلى الكائن DataGridView، لنقم الآن بتكبير النموذج ليشمل معظم الشاشة، وفي نافذة مصادر البيانات نختار الجدول Instructors ونختار السهم الصغير بجانبه لترى قائمة من الكائنات التي تستطيع أن ترتبط مع الجدول كما في هذه الصورة:



الخيارات المتوفرة لعرض جدول من قاعدة البيانات هي:

DataGridView لعرض الجدول بداخل الكائن DataGridView على شكل جدول يحتوي على صفوف وأعمدة.

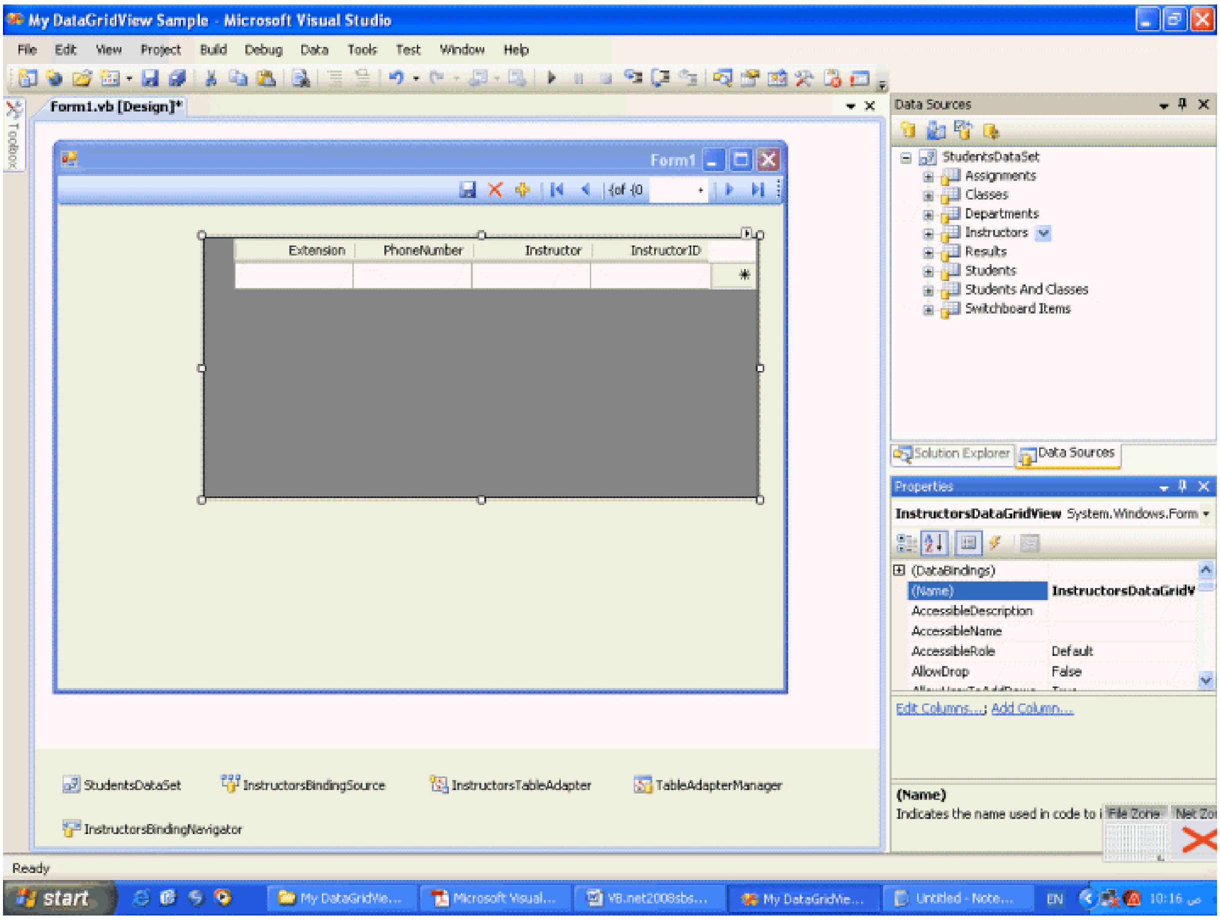
Details ويستخدم لعرض الجدول على شكل مفصل على النموذج حيث يقوم بعرض صندوق نصي لكل حقل وبجانبه ليبل تعريفي بصندوق النص.

None الخيار الثالث يمنع ربط هذا الجدول مع أية كائن، إذا اخترنا هذا الخيار فلن يمكننا ربط الجدول مع الكائنات.

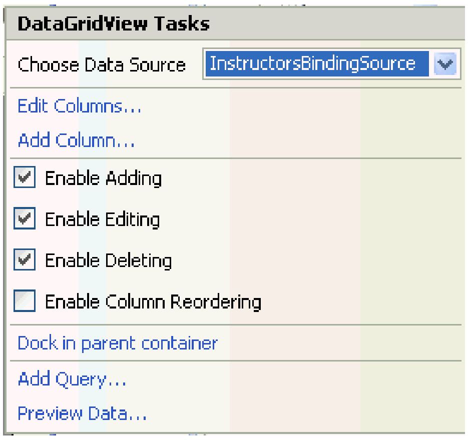
Customize يمكننا من اختيار كائن جديد غير موجود في الخيارات ولابد أن يتقبل هذا الكائن عرض محتويات جدول بأكمله.

٧-ننقر على DataGridView ونقوم بإضافتها إلى أعلى يمين النموذج، سيقوم الفيجوال بيسك بإضافة المكونات StudentsDataSet و InstructorsBindingSource و InstructorsTableAdapter و TableAdapterManager و InstructorsBindingNavigator و كذلك الكائن InstructorsDataGridView

مباشرة إلى التطبيق وسيكون التطبيق كما في هذه الصورة:

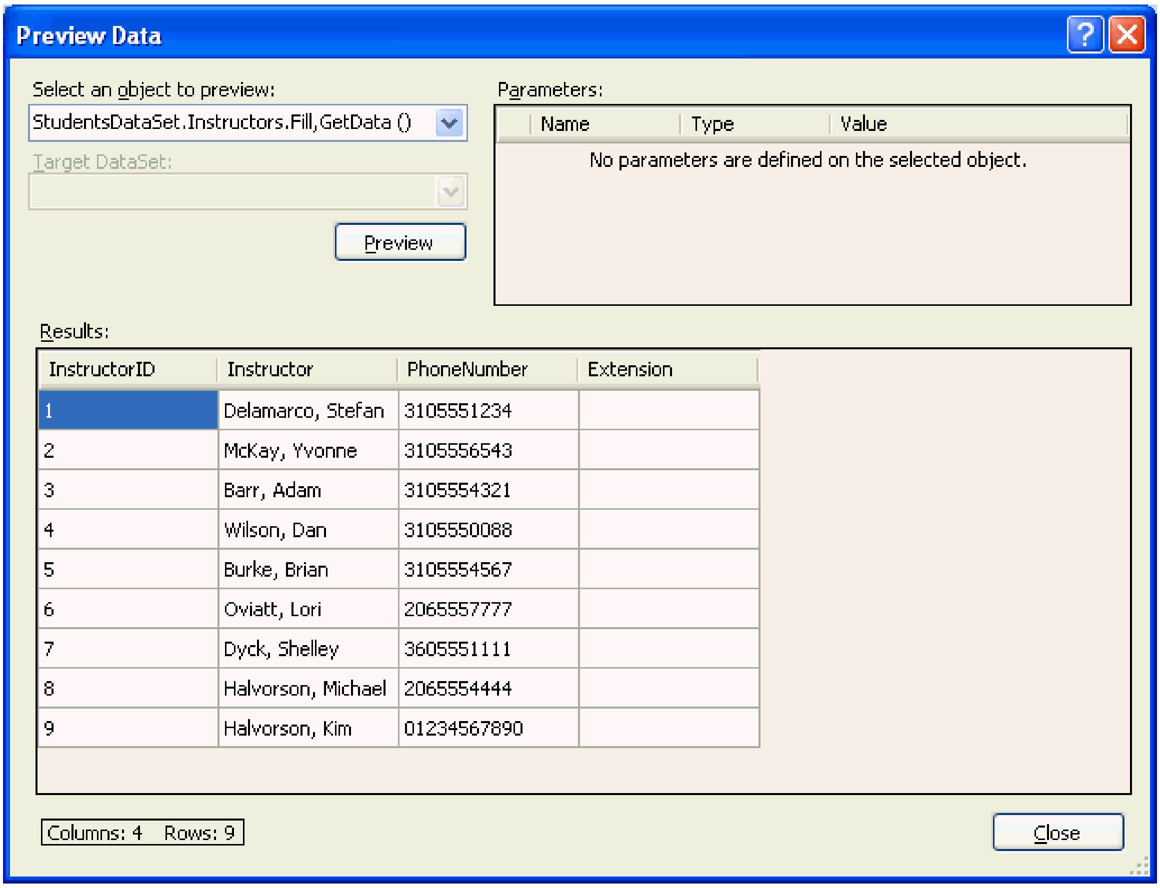


٨-ستلاحظ عدم وجود بيانات على الــ DataGrid في هذه اللحظة وسيقوم البرنامج بتحميل البيانات بعد تنفيذه، اذهب الآن إلى السهم الصغير أعلى يمين الــ DataGrid ونضغط عليه لتظهر لنا النافذة التالية:



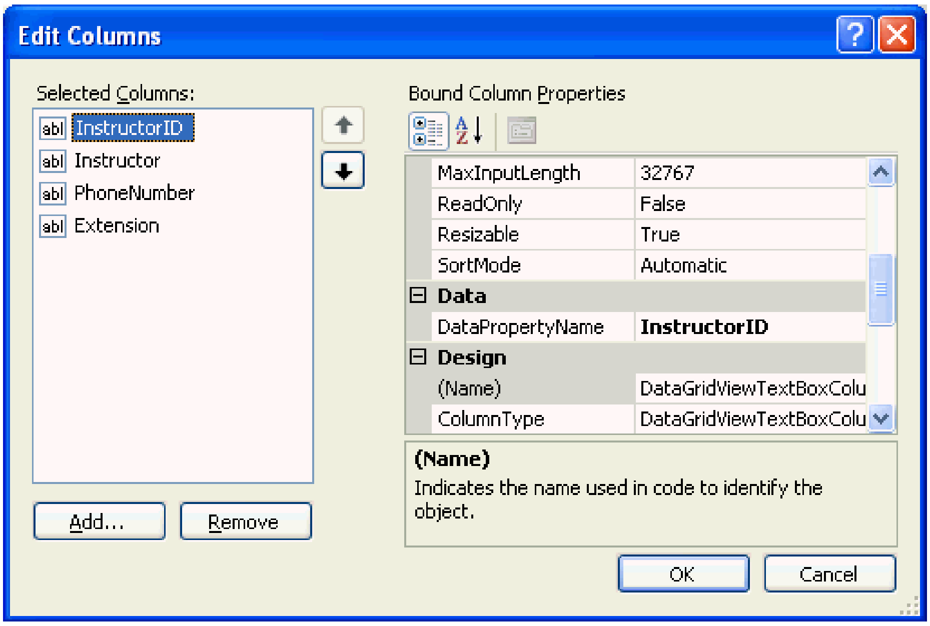
حيث تسمح بتعديل خيارات الــ DataGrid بالسماح للمستخدم بالإضافة و التعديل و الحذف

على التوالي، كذلك بعض الخيارات للـــ DataGrid مثل خيار Dock وهو الالتصاق بجانبين من النموذج حتى إذا قام المستخدم بتكبير النموذج تلصق الـــ DataGrid بالجانبين اليمين والأعلى أو اليسار والأعلى حسب اتجاه النموذج، يمكننا كذلك من استعراض البيانات في مرحلة تصميم البرنامج وبدون الحاجة لتنفيذ البرنامج وذلك من الخيار الأخير Preview Data، نقوم بالضغط على الخيار الأخير لتظهر لنا نافذة جديدة نختار فيها Preview لتقوم بعرض البيانات كما في هذه الصورة:

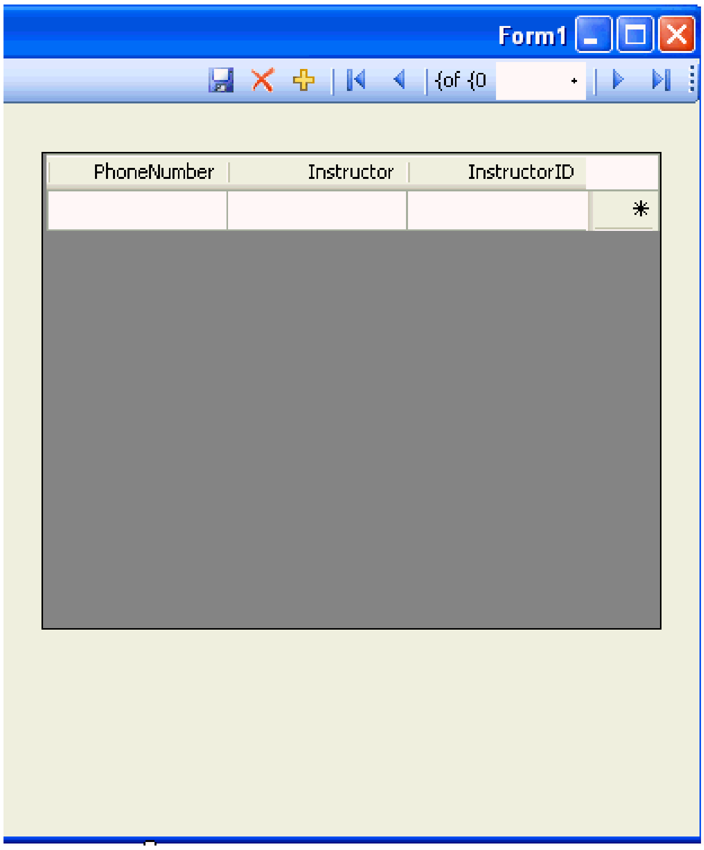


ستلاحظ العمود الرابع تحت اسم Extension فارغ ولا يحتوي على بيانات وقد يسبب الحيرة عند مستخدم البيانات لذلك نقوم بحذفه في الخطوة التالية.

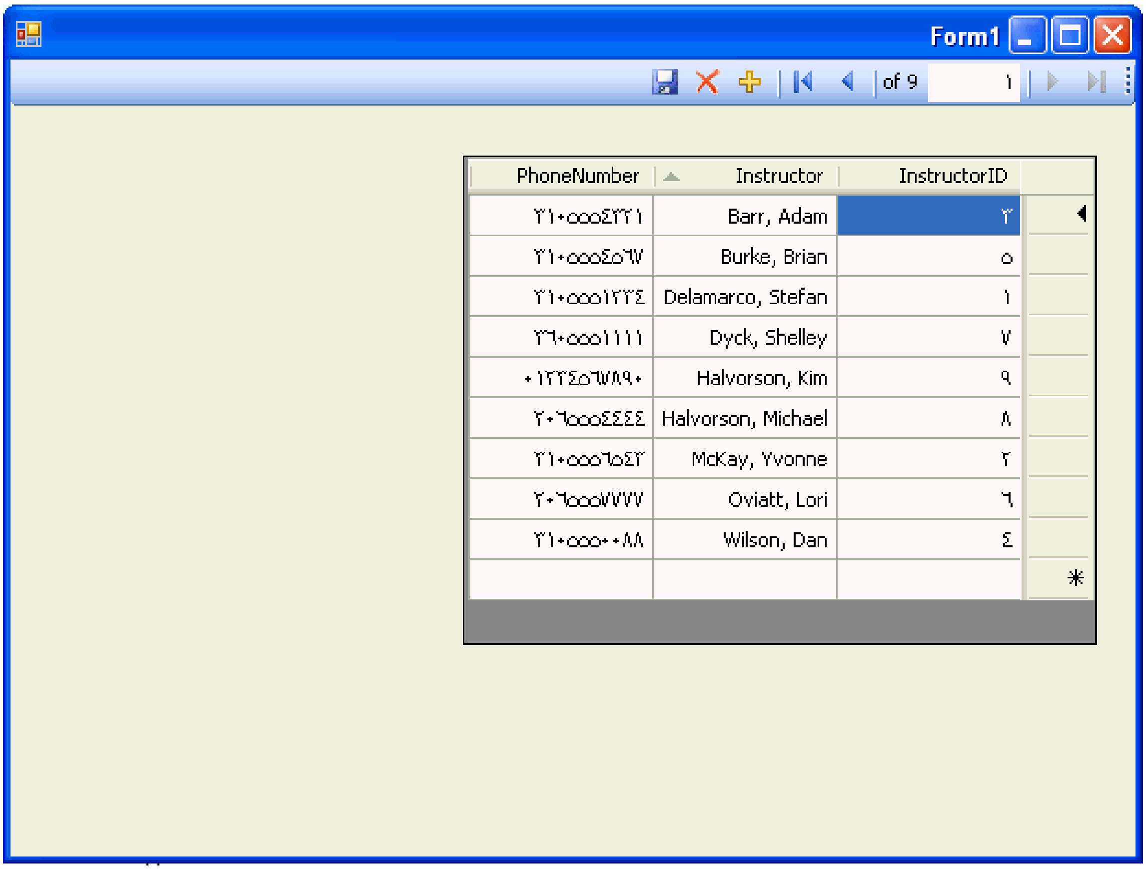
٩- نقوم بإغلاق النافذة السابقة ونذهب إلى النافذة التي قبلها لنختار منها Edit Columns لتظهر لنا هذه النافذة الجديدة:



من هذه النافذة نختار Extension ثم نضغط على الزر Remove ثم نضغط على الزر Ok، بعد التعديل سيكون النموذج كما في هذه الصورة:



١٠- نقوم الآن بتنفيذ البرنامج، مباشرة بعد تنفيذ البرنامج سيقوم البرنامج بتعبئة البيانات في الـــ DataGrid لتظهر كما في الصورة التالية:



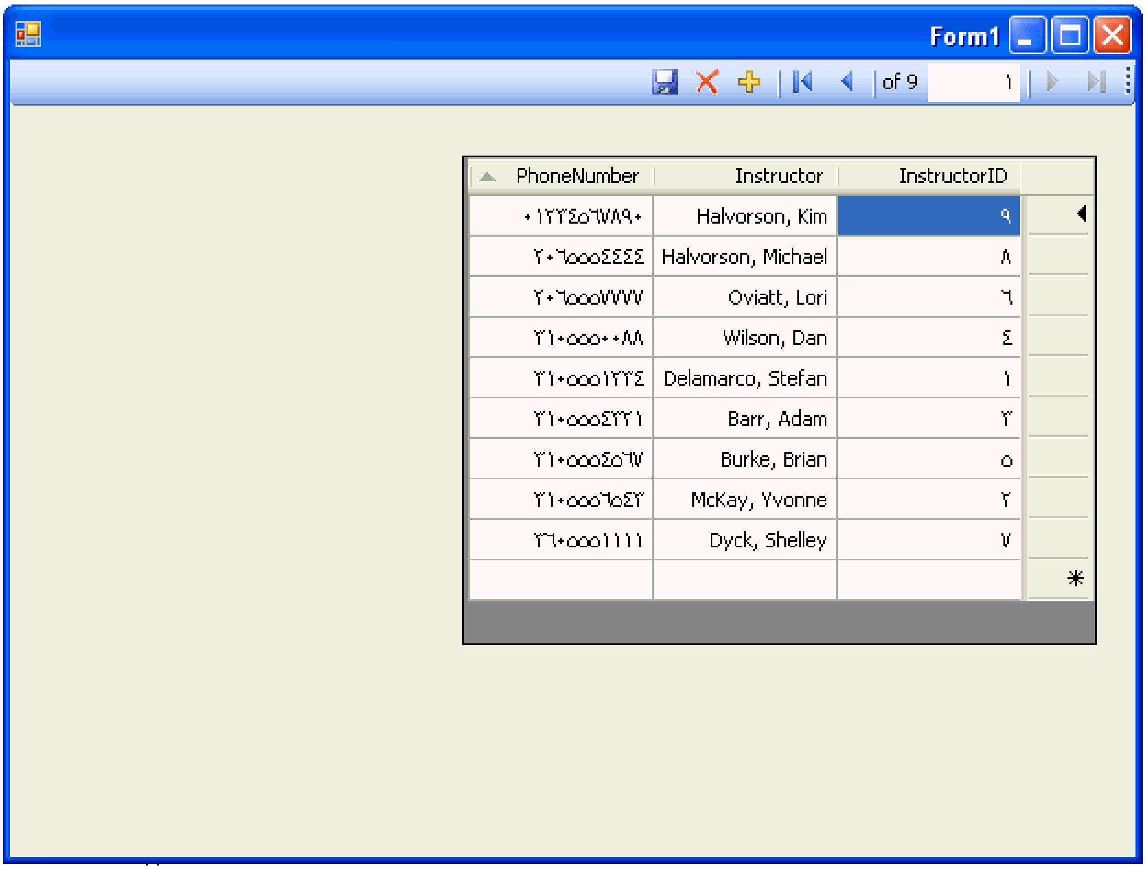
كيف يقوم البرنامج بفعل ذلك بعد تنفيذ البرنامج، نراجع منطقة الكود عند الحدث تحميل النموذج Form1_Load لنرى هذا الكود وهو الذي يقوم بتعبئة البيانات:

```
Me.InstructorsTableAdapter.Fill(Me.StudentsDataSet.Instructors)
```

هذا السطر البرمجي قام الفيجوال بيسك بإضافته بشكل أوتوماتكي للكود عندما قمت بسحب الجدول Instructors إلى النموذج.

١١- بعد تنفيذ برنامجك قد تلاحظ فرق بسيط بين صورة نافذة برنامجي وصورة نافذة برنامجك حيث الجدول عندي من اليمين إلى اليسار وكذلك الأرقام باللغة العربية وهذا يعود إلى الخيار Right-to-Lift من خيارات النموذج حيث قمت بضبطه على True، هذا الخيار يقوم بتحويل النموذج ومحتوياته من اليمين إلى اليسار ليلاءم المستخدم العربي.

١٢- بعد عرض البيانات تستطيع أن تذهب بالماوس إلى المكان الفاصل بين العمود والذي يليه لتقوم بتكبير عمود معين لإن خيار السماح بالتكبير AllowUserToResizeColumns مضبوط على True وتستطيع أن تمنع المستخدم من ذلك بضبط الخيار على False، تستطيع كذلك ترتيب محتويات عمود معين تصاعدياً أو تنازليا بالنقر على اسم العمود في الأعلى، فبالضغط على العمود PhoneNumber في الأعلى يقوم البرنامج بترتيب أرقام الهواتف بشكل تصاعدي من الأصغر إلى الأكبر كما في هذه الصورة:



انظر المثلث الصغير في العمود PhoneNumber والذي يبينُ لنا أن الترتيب في الجدول على أساس هذا العمود وبشكل تصاعدي، إذا قمنا بالنقر مرة أُخرى على نفس العمود سيقوم بعكس الترتيب إلى الترتيب التنازلي، جرب أن تنقر على بقية الأعمدة أكثر من مرة لتشاهد كيف تتم عملية ترتيب الجدول بناءاً على العمود الذي تقوم بنقره تصاعدياً أو تنازلياً وكيف

لهذه الميزة أن تسهل على المستخدم في وقت استعراض الجداول التي تحتوي على البيانات الكثيرة. بعد تجربة تكبير الأعمدة وترتيب محتوياتها قم الآن بإغلاق البرنامج.

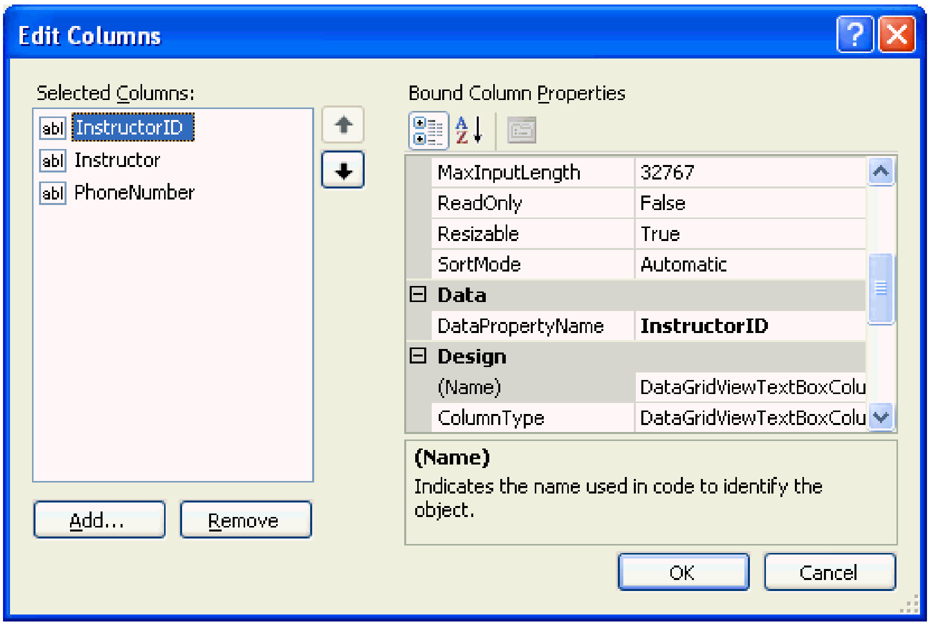
تغيير شكل خلايا الجدول DataGrid Formatting

لتغيير شكل خلايا الــ DataGrid سواءاً مساحة الخلايا أو لون الخلفية في الخلايا وغيرها من الخصائص نتبع التالي:

١- نواصل مع المثال السابق ونستعرض النموذج في وقت التصميم ونختار الجدول DataGrid على النموذج.

٢- نذهب إلى الخاصية Columns التابعة للــ DataGrid ونضغط على الزر ... ليفتح لنا نافذة من خلالها نقوم بالتعديلات التي نريدها كما يلي.

٣- في هذه النافذة:



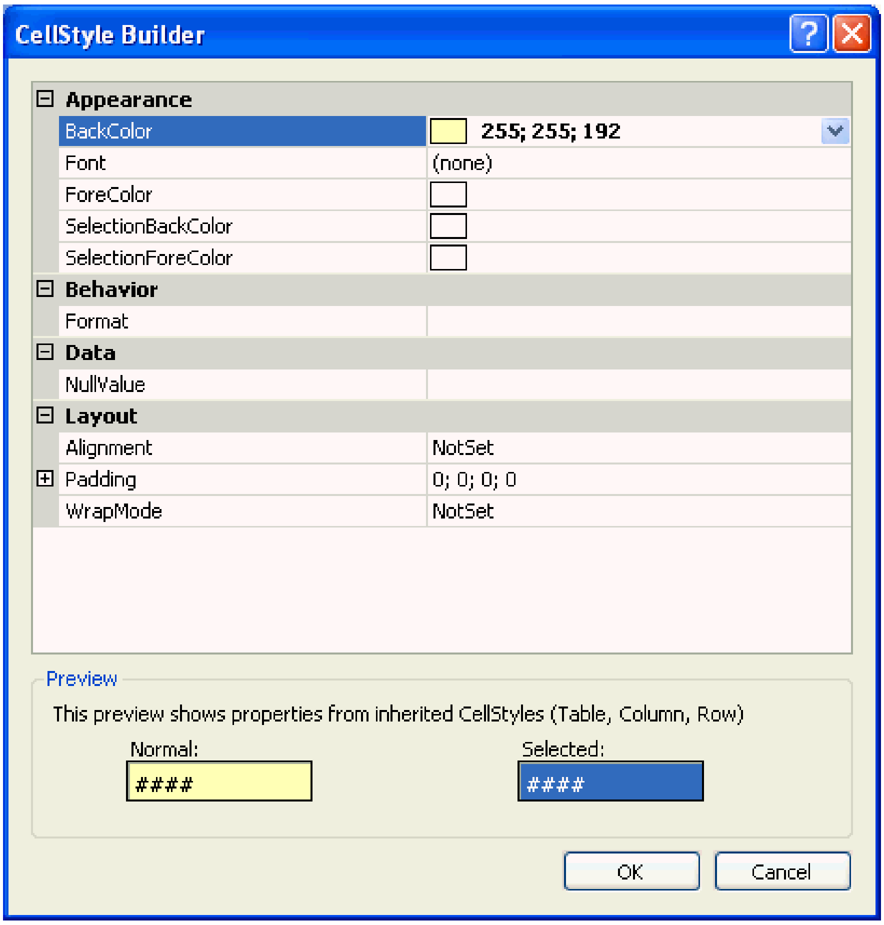
تذكر لقد استخدمنا هذه النافذة لحذف العمود الفارغ سابقاً، الآن سنقوم بالتعديلات المطلوبة على الخلايا، العمود الذي نقوم بالتعديل عليه هو العمود المظلل على يسار النافذة فنبدأ بالتعديل ليقوم بتعديل العمود InstructorID وإذا أردنا تعديل عمود آخر نختاره من يسار

النافذة، نبدأ الآن مع العمود InstructorID نضبط الخاصية Width (عرض العمود) على ٧٠ وتقاس المسافة بالنقطة Pixel، ثم نضغط Ok.

٤- نعود لنافذة الخصائص Properties ونضبط الخاصية ColumnHeadersVisible على False، هذه الخاصية ستحذف أسماء الأعمدة من أعلى الجدول DataGrid.

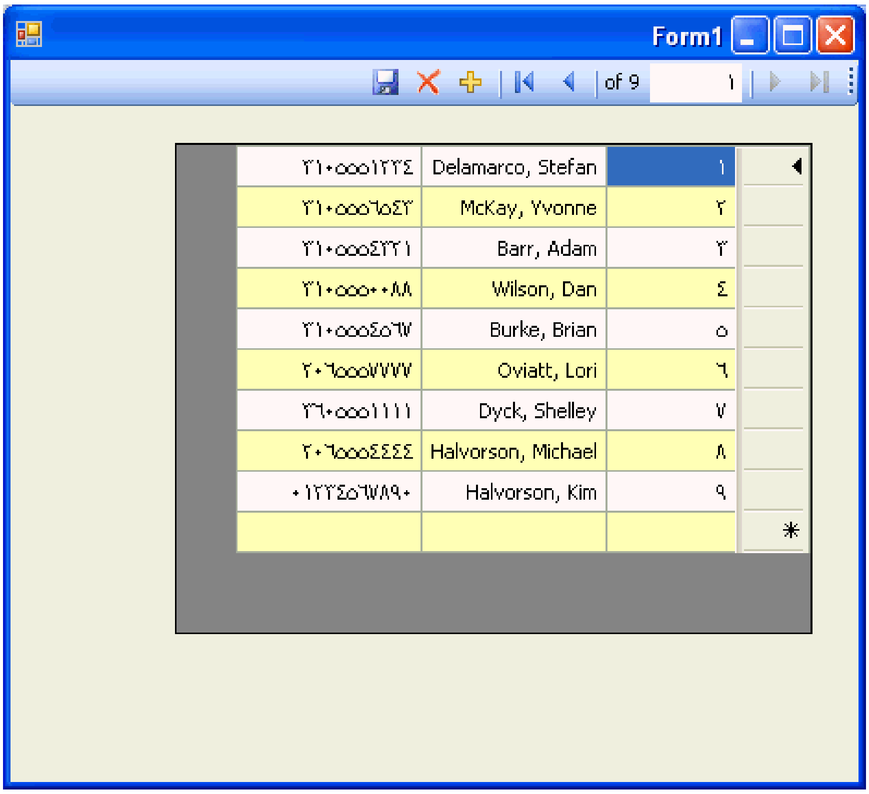
٥- نذهب إلى الخاصية AlternatingRowsDefaultCellStyle ونضغط على الزر ...، هذه الخاصية تتحكم في لون الصفوف عند التنقل من صف إلى آخر، تغيير هذه الخاصية يقوم بتغيير لون الصفوف إلى لونين أحدهما أبيض والآخر اللون الذي تختاره أنت، هذا التغيير مفيد إذا كانت البيانات كثيرة حيث تُسهل عملية قراءة البيانات.

٦- نختار الخاصية BackColor ونختار منها Custom ومنها نختار اللون الأصفر الفاتح، ستكون النافذة كما في هذا الشكل:



نضغط على Ok للموافقة.

٧- الآن عند تنفيذ البرنامج سنرى صفوف البيانات بلونين أحدهما أبيض والآخر أصفر فاتح، نقوم الآن بتنفيذ البرنامج، سيظهر البرنامج كما في هذا الشكل:



٨- تستطيع الغوص في نافذة الخصائص التابعة للـ DataGrid لتجد خصائص أُخرى ممتعه، ملاحظة: التطبيق متوفر في المرفق رقم ٠٥٨.

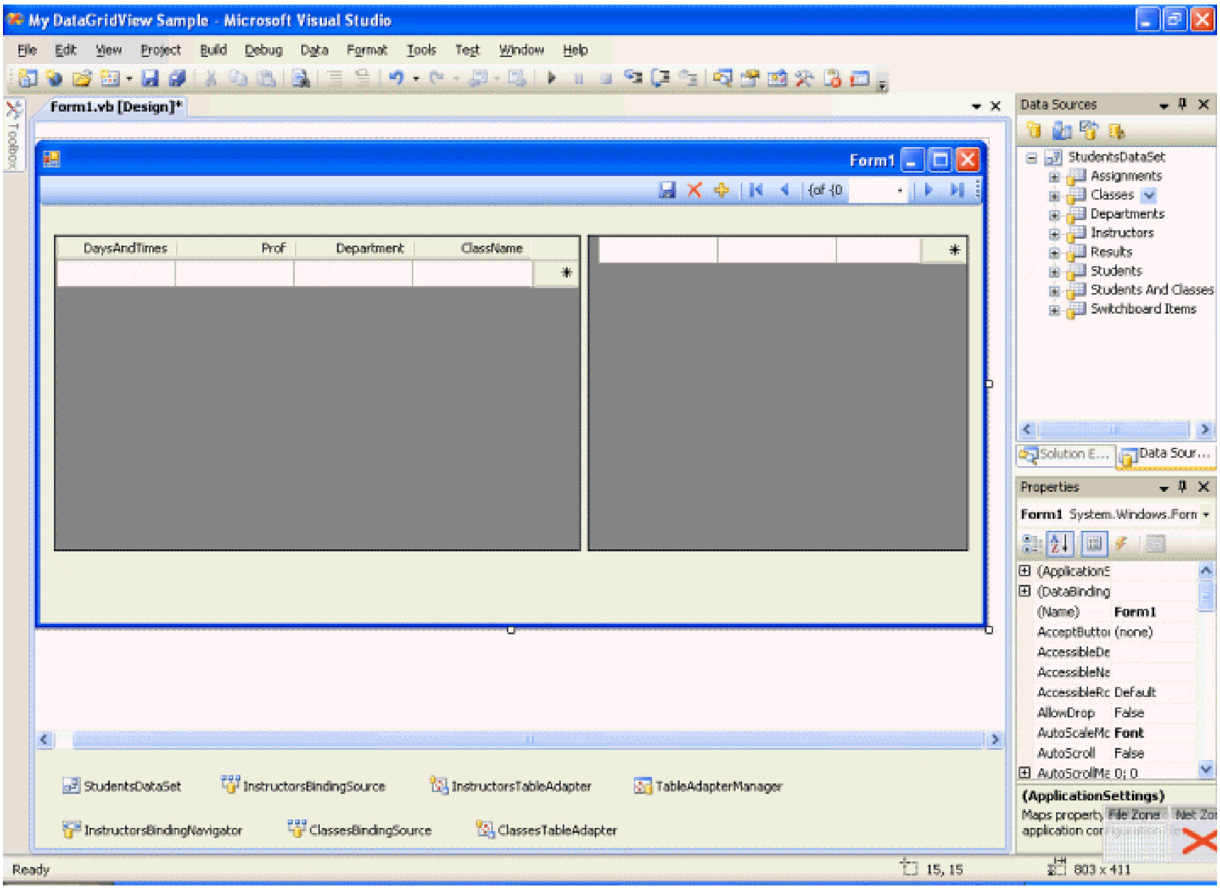
التعامل قواعد البيانات الكبيرة: إضافة DataGrid ثانية وكذلك مستعرض إضافي

عند التعامل مع قواعد البيانات الكبيرة التي تحتوي على أكثر من جدول، قد نحتاج إلى إضافة DataGrid أُخرى للنموذج وإضافة مستعرض جديد للـ DataGrid، إضافة الـ DataGrid من مصادر البيانات سهل، الإضافة الجيدة هنا هي إضافة مستعرض جديد وتعديل الخاصية BindingSource لعرض البيانات، لنأخذ مثالاً على ذلك:

١- بالاستفادة من التطبيق في المثال السابق، نقوم بإضافة الجدول Classes (من نافذة مصادر البيانات) إلى النموذج (الفورم)، سيقوم الفيجوال بيسك بإضافة DataGrid جديدة إلى النموذج تحت اسم ClassesDataGridView.

٢-نضغط Right-Click على ClassesDataGridView ونختار Edit Columns، نقوم باختيار الأعمدة التالية ClassID, SectionNumber, Term, Units, Year, Location, Notes ونختار Remove لكل عمود على حده بعد ذلك نضغط Ok، قمنا بإلغاء الأعمدة المذكورة لإن النموذج لا يتسع لها.

٣-نقوم بتعديل مساحة النموذج ليتسع للجدولين وتكون جميع الأعمدة فيهما واضحة وغير مخفية، كما في هذه الصورة:



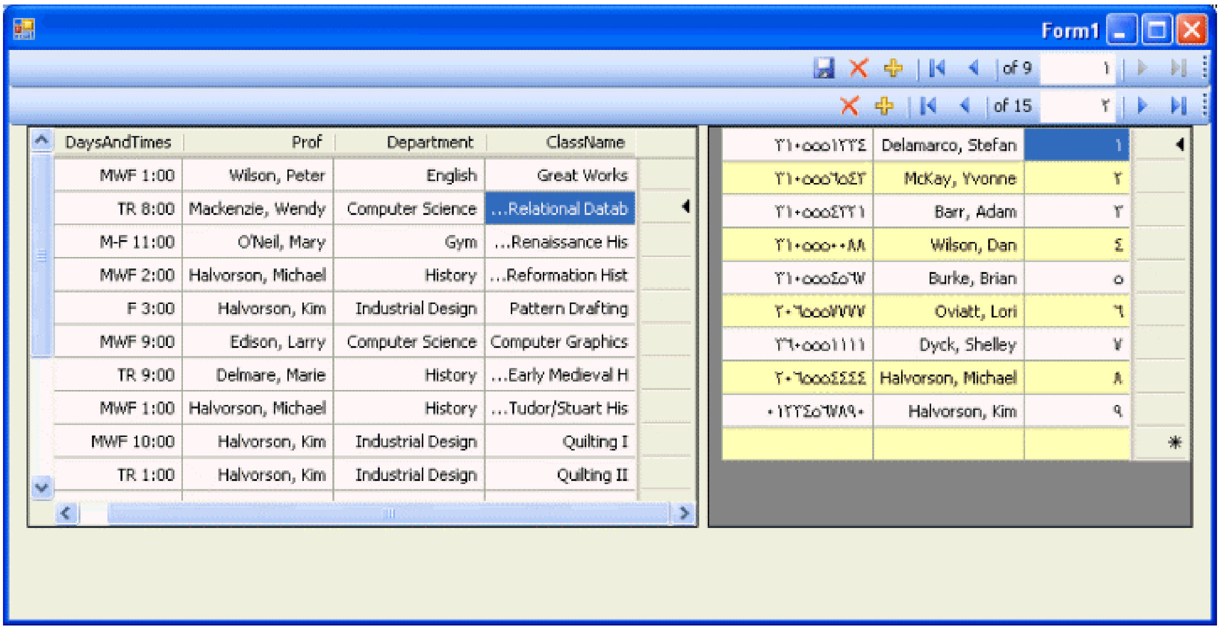
نلاحظ أن الفيجوال بيسك قد أضاف مكونين جديد في الأسفل وهما ClassesBindingSource و ClassesTableAdapter، هذين المكونين هما عبارة عن وسطاء لنقل البيانات بين الجدول ClassesDataGridView وبين قاعدة البيانات.

٤- نقوم الآن بإضافة مستعرض ثاني للبيانات في الجدول ClassesDataGridView لإن المستعرض الأول InstructorsBindingNavigator يقوم فقط باستعراض البيانات في الجدول InstructorsDataGridView، نذهب إلى صندوق الأدوات وتحت التبويب Data ننقر نقرتين على الأداة BindingNavigator، يقوم الفيجوال بيسك بإضافة مكون جديد أسفل النموذج في شريط المكونات باسم BindingNavigator1 ويضيف شريط استعراض للبيانات في النموذج أسفل الشريط السابق.

٥- من نافذة الخصائص نقوم بتغيير الخاصية Name للشريط الجديد إلى ClassesBindingNavigator، حتى تسهل علينا مراجعة الكود.

٦- نقوم بضبط الخاصية BindingSource على ClassesBindingSource، سيقدم لنا الفيجوال بيسك خيارين نختار منهما ClassesBindingSource، الآن تم الربط بين شريط استعراض البيانات الثاني والجدول الثاني ClassesDataGridView.

٧- نقوم بحفظ التطبيق ثم تنفيذه، وبعد تنفيذ برنامجنا سنشاهد تطبيق يحتوي على جدولين ومستعرضين، كما في الصورة:



٨- جرب الآن التنقل بين البيانات باستخدام المستعرضين، يا سلام يقوم البرنامج بالتنقل بين البيانات، استخدم أحد المستعرضين في الذهاب إلى حقل معين من البيانات ثم استخدم الآخر

في الذهاب غلى حقل آخر، رائع جداً، بالطبع يعمل المستعرضين بشكل مستقل عن بعضهما البعض، هذا سيعتبر مفيد إذا كان لدينا جدولين يحتويان على بيانات كثيرة ونريد مقارنتهما يدوياً.

٩-قم الآن بإغلاق التطبيق وتذكر بأنه توجد نسخة من التطبيق في المرفق رقم ٠٥٩.

خطوة إضافية للأمام: الحفظ في قاعدة البيانات

كما وضحنا سابقاً بأن مجموعة البيانات Dataset لا تعبر عن قاعدة البيانات الأصلية بشكل مباشر وإنما تقوم بأخذ البيانات من قاعدة البيانات ثم عرضها على المستخدم وإذا اخترنا حفظ يقوم البرنامج بحفظ التعديلات مؤقتاً، ينتهي هذا الحفظ عند إغلاق البرنامج وعند فتحه مرة أُخرى نفتح البيانات القديمة من قاعدة البيانات، ولإعتماد التعديلات التي يقوم بها المستخدم ونقلها إلى قاعدة البيانات يجب أن نضبط الخاصية ReadOnly التابعة للـــ DataGrid على False ثم التعديل على الــ DataGrid وبعد التعديل نضغط على زر الحفظ في شريط مستعرض البيانات، في المثال التالي سنتعلم كيف نفعل ذلك:

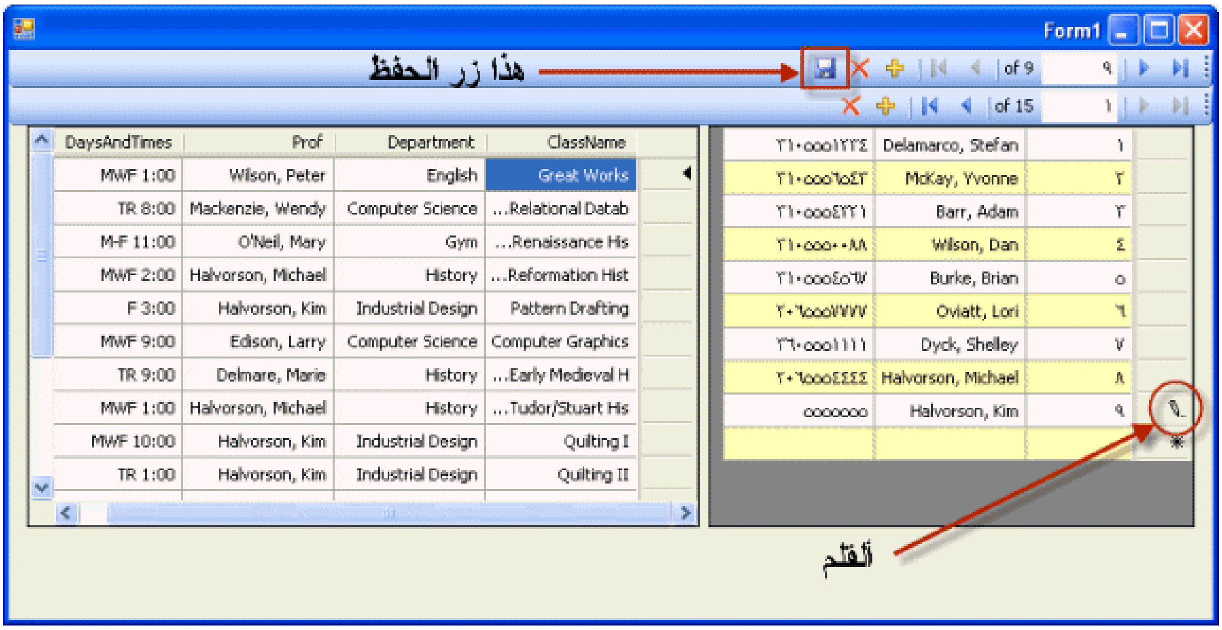
١-نفتح المشروع السابق (متوفر نسخة منه في المرفق رقم ٠٥٩).

٢-نختار الجدول InstructorsDataGridView، ونذهب إلى نافذة الخصائص.

٣- نذهب إلى الخاصية ReadOnly، الخاصية مضبوطة على False يعني أن المستخدم إذا قام بالتعديل يمكنه ذلك وإذا قام بحفظ التعديلات تنتقل هذه التعديلات إلى قاعدة البيانات، أما إذا قمنا بضبط هذه الخاصية على True فلن يستطيع المستخدم التعديل على الــ DataGrid ومن ثم لن يتم حفظ أية تعديلات. الآن سنتعمد الخاصية ReadOnly على ما هي علية False لكي نسمح للمستخدم بالتعديل على البيانات.

٤-نقوم بتنفيذ البرنامج، وبعد ظهور نافذة البرنامج وعليها الجدولين نختار من النافذة الأولى الخلية التي تحوي الاسم Halvorson, Kim وهي الخلية الأخيرة ونقوم بكتابة الرقم التالي فيها: "٥٥٥٥٥٥٥" أو أي رقم آخر، عندما نقوم بالتعديل في الخلية المذكورة يظهر قلم رصاص صغير مقابل الخلية ليُعلمنا بأننا نقوم بعملية تعديل للبيانات كما في هذه

الصورة:



بعد تكملة الكتابة الخلية نضغط على زر الحفظ الموضح بالصورة، بعد الضغط على زر الحفظ يقوم البرنامج بحفظ التغيرات في قاعدة البيانات ويمكنك التأكد من ذلك بإغلاق البرنامج ثم فتحه مرة أُخرى لتشاهد البيانات التي قمت بتغييرها قد تغيرت، ماذا حدث ما هو كود الحفظ في قاعدة البيانات، نذهب إلى زر الحفظ في مرحلة تصميم البرنامج وننقر علية نقرتين لينقلنا إلى منطقة الكود وإلى الحدث Click وسنرى هذا الكود وهو الذي يقوم بحفظ البيانات في قاعدة البيانات:

```
Me.TableAdapterManager.UpdateAll(Me.StudentsDataSet)
```

هي عبارة عن مكون جديد في فيجوال بيسك ٢٠٠٨ يسمى TableAdapterManagerالأداة فتقوم بحفظ UpdateAllللك بالتعامل مع أكثر من قاعدة بيانات من برنامجك، أما الطريقة التعديلات على كل الجداول المفتوحة في برنامجك، هذا يعني أنها لا تحفظ التعديلات في الجدول Classes فقط وإنما تقوم بحفظ التعديلات التي حدثت حتى في الجدول Instructors ، إذا أردت فقط استخدم هذا الكود:Instructorsفقط حفظ التعديلات على الجدول

```
Me.InstructorsTableAdapter.Update(Me.StudentsDataSet.Instructors)
```

والذي يقوم كان الفيجوال ٢٠٠٥ يقوم بكتابته فقط بدون كود حفظ كل الجداول كما في ٢٠٠٨، لكننا الآن نستطيع كتابة الكود للجدول الذي نريد حفظ تعديلاته فقط، تستطيع الآن حفظ التعديلات التي تكتبها على الجداول تستطيع كذلك إضافة سجلات جديدة لقواعد البيانات في الجداول بالذهاب إلى تحت آخر سجل والكتابة فيه ثم حفظ التعديلات، إذا أردنا أن نمنع المستخدم من التعديل على الجداول نضبط الخاصية ReadOnly للجدول DataGrid على True.

تستطيع القول بأنك وضعت قدمك على الخطوات الأولى في تعلم التعامل مع قواعد البيانات، تستطيع مواصلة تقدمك بالرجوع إلى الكتب المتخصصة في قواعد البيانات والرجوع إلى المصادر التعليمية على إنترنت مثل موقع فيجوال بيسك للعرب وموقع الفريق العربي للبرمجة.

خلاصة الفصل التاسع عشر

من اجل أن	قم بالتالي
تتواصل مع الجداول في قواعد البيانات	من قائمة Data نختار Add New Data Source ومنها نختار قاعدة البيانات ونختار منها ما نحتاج من الجداول والحقول لنكون ما يسمى بمجموعة البيانات Dataset، ومن نافذة مصادر البيانات Data Sources نختار الحقول التي نريدها ونضعها على النموذج.
وضع الأداة "الجدول" DataGrid ووضع بيانات جدول كامل عليها	من نافذة مصادر البيانات Data Sources بجانب مستكشف المشروع نختار أيقونة الجدول ونسحبها بالماوس ونضعها على النموذج، نقوم بتكبيرها لتكون كل الأعمدة ظاهرة لنا.
عرض بيانات التي سيقوم التطبيق بعرضها على الجدول	ننقر فوق الجدول DataGrid وسنرى في أعلى يمين الجدول سهم صغير ننقر علية لتظهر لنا قائمة نختار منها Preview Data لتظهر لنا نافذة نضغط على الزر Preview لنرى البيانات التي ستعرض على الجدول

DataGrid	DataGrid بعد تنفيذ البرنامج.
حذف عمود من الجدول DataGrid	نضغط على السهم الصغير المذكور في الخطوة السابقة ونختار Edit Columns ثم نختار العمود الذي نريد إلغاؤه ونختار Remove
ترتيب السجلات في الجدول	في لحظة تشغيل البرنامج نذهب إلى العمود الذي نريد ترتيب السجلات على أساسه، وعند اسم العمود ننقر ليقوم البرنامج بترتيب الجدول على أساسه ترتيباً تصاعدياً ننقر مرة أُخرى لنقوم بالترتيب بطريقة تنازليه.
تغيير عرض عمود معين في الجدول	من نافذة الخصائص نختار الخاصية Columns وننقر الزر ... ومن النافذة التي تظهر نعدل قيمة الخاصية Width بالقيمة المناسبة.
إخفاء أسماء الأعمدة في الجدول	نضبط الخاصية ColumnHeadersVisible على False
تغيير ألوان السجلات الفردية عن الزوجية في الجدول	نقوم بتغيير الألوان بهذا الشكل لنجعل قراءة الجدول سهله ومريحة، وذلك بالذهاب إلى الخاصية AlternatingRowsDefaultCellStyle وفي النافذة التي تظهر نعدل الخاصية BackColor إلى اللون الذي نريد، ليقوم البرنامج باعتماده للسجلات الزوجية،واللون الأبيض للفردية.
تغيير لون خط الجدول	من الخاصية GridColor
إضافة جدول آخر إلى DataGrid النموذج	بواسطة الماوس من نافذة مصادر البيانات Data Sources نختار الجدول الذي نريده ونسحبه إلى النموذج، نقوم بتكبير النموذج ليتسع للجدول الجديد وإذا أردنا إضافة مستعرض للجدول نضيف الكائن BindingNavigator من صندوق الأدوات ونضبط الخاصية BindingSource له على الجدول الجديد.

منع المستخدم من التعديل على البيانات في الـــ DataGrid	نضبط الخاصية ReadOnly لـــ DataGrid على True.
حفظ التعديلات التي يقوم بها المستخدم على قاعدة البيانات	أولاً لابد من ضبط الخاصية ReadOnly للجدول DataGrid على False ثم في وقت تنفيذ البرنامج نضغط على الزر حفظ في مستعرض البيانات، ويمكنك استخدام الكود التالي في حفظ التعديلات إلى قاعدة البيانات: Me.TableAdapterManager.UpdateAll method

الفصل العشرون

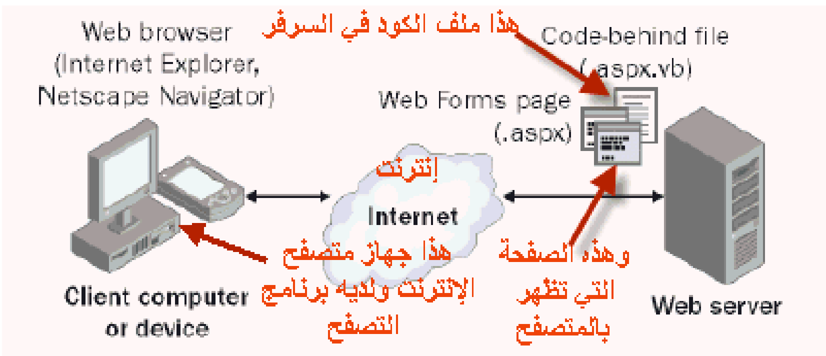
إنشاء مواقع الإنترنت وصفحات الإنترنت بواسطة Visual Web Developer و ASP.NET

تمت إضافة الأداة Visual Web Developer إلى نسخة الفيجوال بيسك ٢٠٠٨ والتي تظهر بنفس مظهر بيئة التطوير ولكنها مخصصة لبرمجة الويب و ASP.NET 3.5، أول ظهور لتقنية الـــ ASP.NET كان مع نسخة الفيجوال بيسك ٢٠٠٢ كبديل لـــ WebClasses و DHTML Page Designer في فيجوال بيسك ٦، إننا لن نستطيع هنا أن نقوم بتعريف كامل لبرمجة الويب لكننا نستطيع أن نقوم بأن برمجة الويب تتشابه مع برمجة تطبيقات الويندوز، حتى إذا كانت لديك خبرة قليلة بالـــ HTML أو حتى إذا كنت لا تملك أي خبرة إقرأ هذا الفصل لتستفيد وتتعلم كيف تقوم ببناء تطبيق للويب وعرض البيانات من قاعدة بيانات مايكروسوفت أكسس على المتصفح.

مع ASP.NET

ASP.NET 3.5 هي آخر إصدار من تقنية ASP.NET من ميكروسوفت للتعامل مع الويب، تم دعم تقنية الـــ AJAX (Asynchronous JavaScript and XML) في هذه الإصدارة، وتمت إضافة مكونات جديدة وكذلك تمت إضافة LinqDataSource التي تسمح للمبرمج من استخدام تقنية الـــ LINQ (Language-Integrated Query)، تتشابه ASP.NET بعض الشيء مع

ضرتها القديمة ASP (Active Server Pages) لكننا نستطيع أن نقول بأن ASP.NET تحتوي على العديد من التطويرات التي لا تُقاوم، التعامل مع ASP.NET يُعني أنك تقوم بتصميم برامج إنترنت وصفحات إنترنت تحتوي على بيانات وتتعامل مع المستخدم، يمكن فتحها بواسطة متصفحات الإنترنت مثل Internet Explorer و FireFox و Opera وغيرها من المتصفحات، بعد التصميم نقوم بحفظ هذه الصفحات على خوادم إنترنت Web Servers لتعمل، نستطيع تشغيل صفحات الإنترنت التي قمنا بتصميمها على جهازنا المحلي للتأكد أن الصفحات تعمل بشكل صحيح ثم نقل الصفحات المصممة إلى خوادم إنترنت Web Servers، لتشغيل الصفحات المصممة على جهازنا نحتاج إلى ما يسمى بـــ IIS إذا كان لدينا نسخة الفيجوال ٢٠٠٢ أو ٢٠٠٣ أما في نسختي ٢٠٠٥ و ٢٠٠٨ فلا نحتاج له ويمكننا تشغيل الصفحات مباشرة، لإنشاء صفحة إنترنت في فيجوال ٢٠٠٨ نضغط على New Web Site في قائمة File أو الاختصار Shift+Alt+N، ثم نستخدم الأداة Visual Web Developer لإنشاء صفحة إنترنت أو أكثر، وعند تصميم صفحات الإنترنت تتكون الصفحة من قسمين قسم التصميم لإضافة أزرار و غيرها من الكائنات وقسم أكواد لكتابة الكود كما في تصميم برامج الويندوز يوجد واجهة للمستخدم ويوجد مكان للكود، يمكن للكود التابع للقسمين السابقين أن يُحفظ في ملف واحد على هيئة aspx، وبشكل عام يتم حفظ الكود التابع للتصميم والواجهة في ملف aspx وحفظ الكود الذي قمنا بكتابته في ملف على هيئة aspx.vb الشكل التالي يوضح كيف يتم ذلك:



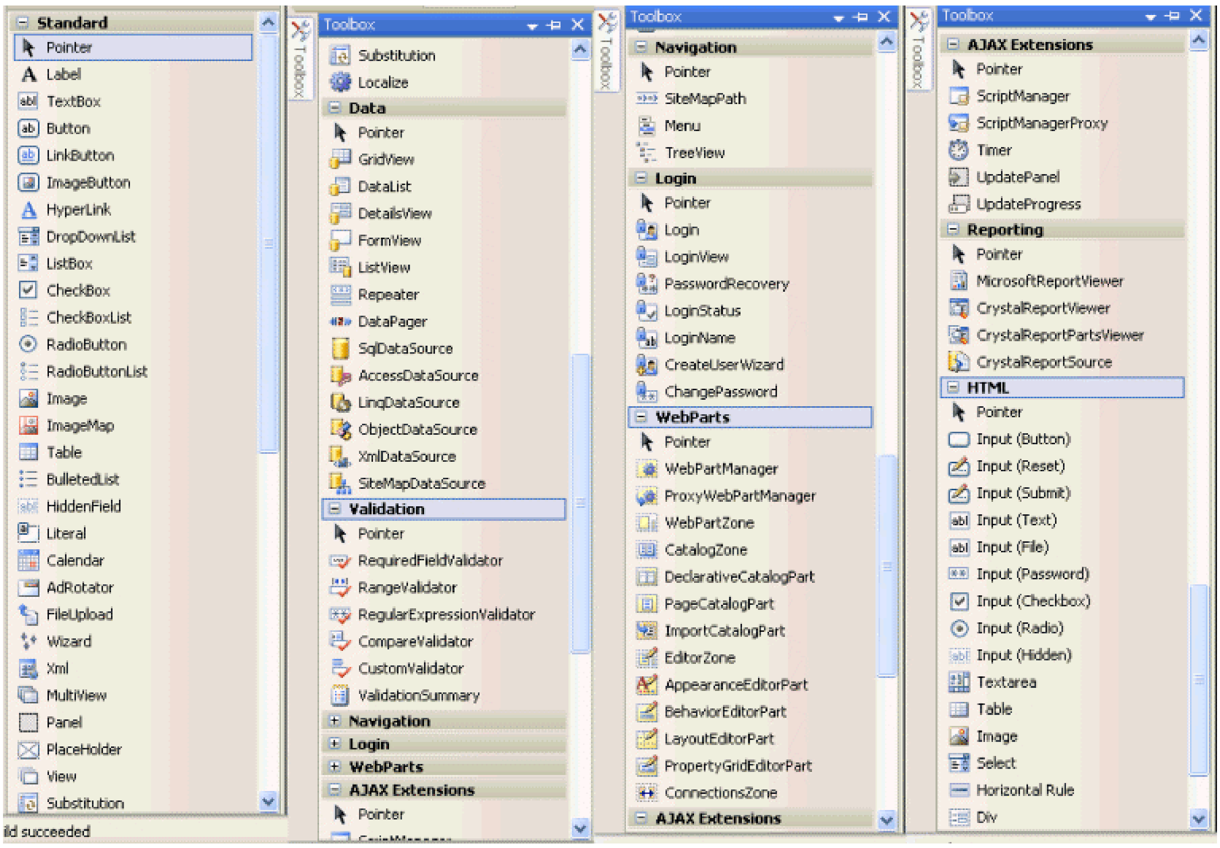
الفرق بين تطبيقات إنترنت وتطبيقات الويندوز

أولاً نقوم باستعراض صفحات الإنترنت بواسطة متصفح لقراءات بيانات وللتعامل مع مواقع الويب، تحتوي صفحات الويب على نفس المحتويات التي تحتويها تطبيقات الويندوز فكليهما

يحتويان على صور، أزرار، نصوص، وبعض الكائنات الأُخرى التي تقدم بيانات ومعلومات، لكن هناك فرق بين الكائنات التي نضيفها إلى صفحات الإنترنت وبين الكائنات على تطبيقات الويندوز، الكائنات التي نضيفها لصفحات الويب توجد في قوائم الـــ Web Developer Toolbox وكل كائن من هذه الكائنات يحتوي على أحداث معينة تختلف عن أحداث برامج الويندوز وكذلك طرق Methods معينة وخصائص مختلفة عن تلك التي تضاف لتطبيقات الويندوز.

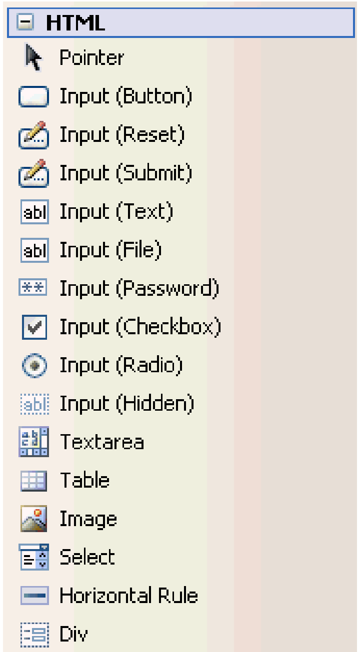
كائنات الويب أو Server Controls

كائنات الويب أو كائنات الخوادم Server Controls هي كائنات التي تضاف إلى تطبيقات الإنترنت وتعتبر أقرب للكائنات المضافة إلى الويندوز من كائنات الـــ HTML، بالإضافة إلى الكائنات المعروفة للتعامل مع تطبيقات الويندوز مثل صناديق النص والأزرار وغيرها من الكائنات هناك كائنات جديدة كلياً مثل الكائن FileUpload الذي يقوم برفع الملف و الكائن LoginView و RequiredFieldValidator في الصورة التالية سنستعرض معظم الكائنات التي تضاف لتطبيقات الويب:



كائنات الــ HTML

كائنات الــ HTML هو الكائنات المرئية القديمة التي تضاف لصفحات الويب وتعتبر مدعومة في كل المتصفحات وتتبع معايير لغة الــ HTML، تحتوي هذه الكائنات على أزرار وصناديق نص وصناديق الاختيار Checkbox، عند استخدام هذه الكائنات فإننا نستخدم أكواد الــ HTML، إذا كانت لديك خبرة سابقة بالــ HTML أو بـ Visual Basic 6 DHTML Page Designer فستكون لك خبرة بمثل هذه الكائنات، هذه المكونات تتميز بأنها مدعومة من كل المتصفحات وسهل جداً التعامل معها لكن عيبها هو المحدودية فليست مرنة بالنسبة للمبرمج ولا تتيح خيارات عديدة للمبرمجين، هذه الكائنات في الصورة التالية:



إنشاء صفحة إنترنت باستخدام Visual Web Developer

قبل برمجة صفحة إنترنت لا بد من معرفة متطلبات البرمجة بالــ ASP.NET والمتطلبات كالتالي:

١- لابد أن يكون على جهازك Visual Web Developer، بحيث يسهل علينا استعراض صفحات الإنترنت المصممة على جهازنا بدون الحاجة لرفعها على خادم إنترنت، في نسخة الفيجوال ٢٠٠٥ و ٢٠٠٨ تستطيع استعراض صفحات الإنترنت المُصَمَمة بإحدى ثلاث طرق:

١- على جهازك المحلي بدون رفعها على الإنترنت.

٢- على موقع Http (على سيرفر يحتوي على IIS والمكونات ذات العلاقة)

٣- على سيرفر Ftp

سنستخدم الخيار الأول في هذا الكتاب لإنه الخيار الأسهل بالنسبة لنا ولا يحتاج سيرفر شخصي ولا برامج إضافية، عند تصميم الموقع في جهازنا المحلي يقوم الفيجوال بيسك بوضع جميع ملفات الموقع في ملف واحد نقوم لاحقاً برفعه على سيرفر إنترنت.

ملاحظة: إذا أردت رفع الصفحات التي تصممها على موقع فتأكد أنك جهازك يحتوي على الـــ IIS والمكونات ذات العلاقة في ويندوز XP نختار إضافة وإزالة البرامج في لوحة التحكم ومنها نختار إضافة أو حذف مكونات الويندوز نختار الـــ IIS ونقوم بتنصيبه على جهازنا،

Control Panel>Add Or Remove Programs>Add/Remove Windows Components>IIS

أما في ويندوز فيستا فنفتح البرامج والميزات في لوحة التحكم ثم نضبط ميزات الويندوز على تفعيل أو إلغاء ثم نختار ميزات الـــ IIS و ASP.NET لتنصيبهما على جهازنا،

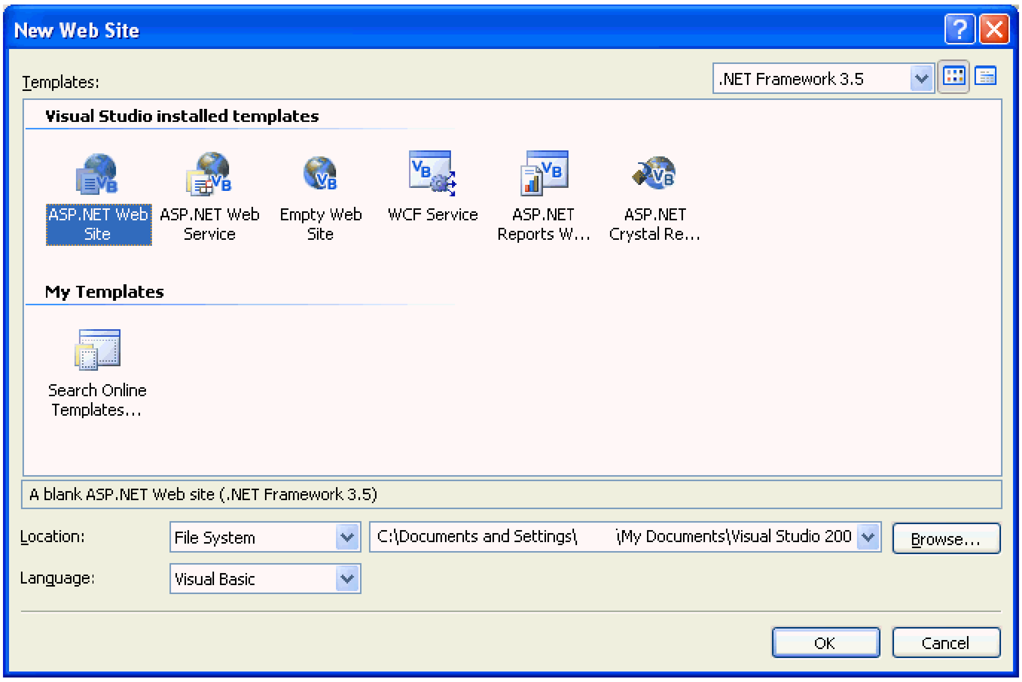
Control Panel>Programs And Features>Turn Windows Features On Or Off>IIS and ASP.NET features

يفضل القيام بهذه الخطوة قبل تنصيب الفيجوال بيسك والدوت نت فريم وورك على جهازنا حتى لا يحدث تعارض لاحقاً، وإذا لم يتم ذلك من قبل وحدث خطأ قد يكون بسبب تنصيب الفيجوال بيسك والدوت نت قبل هذه الخطوة.

مثال تطبيقي على إنشاء صفحة إنترنت

١- نفتح الفيجوال ٢٠٠٨ ومن قائمة File نختار New Web Site أو الاختصار Shift+Alt+N (إذا لم تجد New Web Site في قائمة File فهذا يُعني أنه ليس لديك الـــ Visual Web Developer).

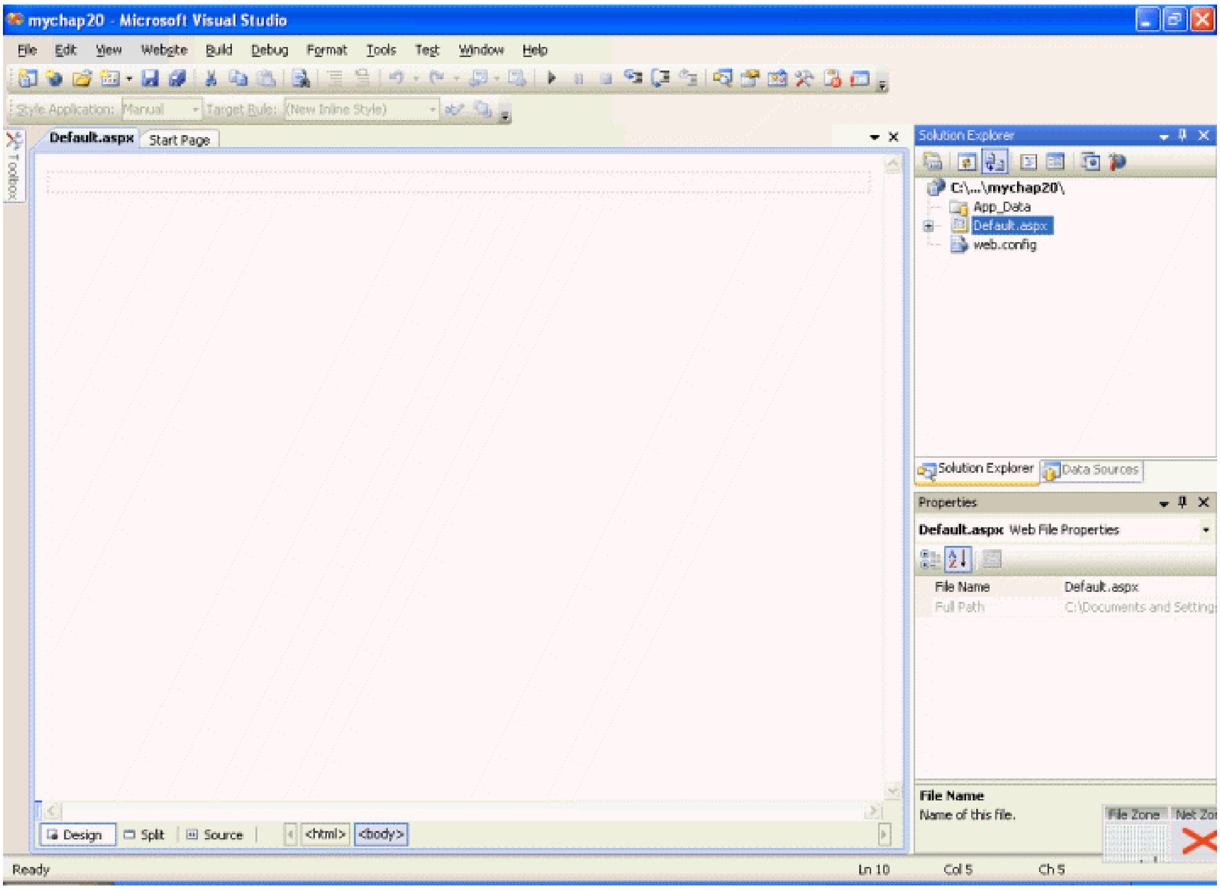
٢-ستظهر لك النافذة التالية:



من النافذة أعلاه سنختار ASP.NET Web Site ونتأكد أن نسخة إطارات العمل هي .Net Framework 3.5 وكذلك لغة البرمجة Visual Basic والخيار Location مضبوط على File System.

٣-نضغط على استعراض Browse لننشئ مجلد جديد باسم mychap20 في المسار c:\vb08sbs أو أي مسار آخر نريده. ستلاحظ الفرق بين تطبيقات الويب وتطبيقات الويندوز بأننا نقوم بحفظ تطبيقات الويب أولاً ثم نصممها بعكس تطبيقات الويندوز التي نُصَممها ثم نحفظها.

٤-نضغط Ok لإنشاء المشروع، يقوم الفيجوال بيسك مباشرة بإنشاء الصفحة Default لتقوم بتصميمها وكذلك كتابة الكود في الملف Default.aspx.vb التابع لها، ستكون بيئة التطوير كما في هذه الصورة:

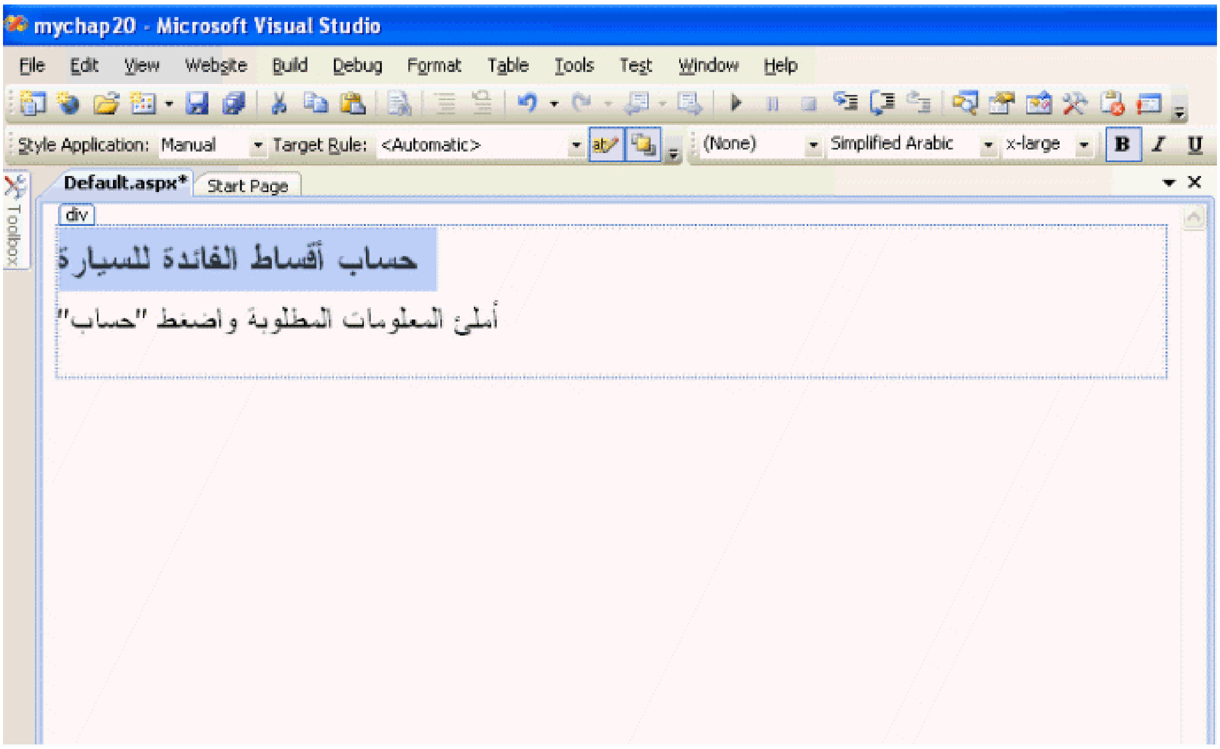


لاحظ في أسفل يسار الصفحة يعرض لنا الفيجوال ٢٠٠٨ ثلاثة خيارات من خيارات التصميم وهي (Design, Split، و Source) يعرض لنا الخيار Design تصميم الصفحة المرئي الذي سوف ظهر في المتصفح، ويعرض الخيار Split مزيجاً بين التصميم والكود أما الخيار Source فيعرض لنا كود الصفحة فقط بصيغة HTML والذي بدوره يظهر في متصفح المستخدم الذي يتصفح صفحتك. في مستكشف المشروع Solution Explorer ستجد العديد من الملفات أولها الصفحة التي تقوم بتصميمها Default.aspx وكذلك صفحة الكود Default.aspx.vb تحتها مباشرة وكذلك ملف الإعدادات Web.config.

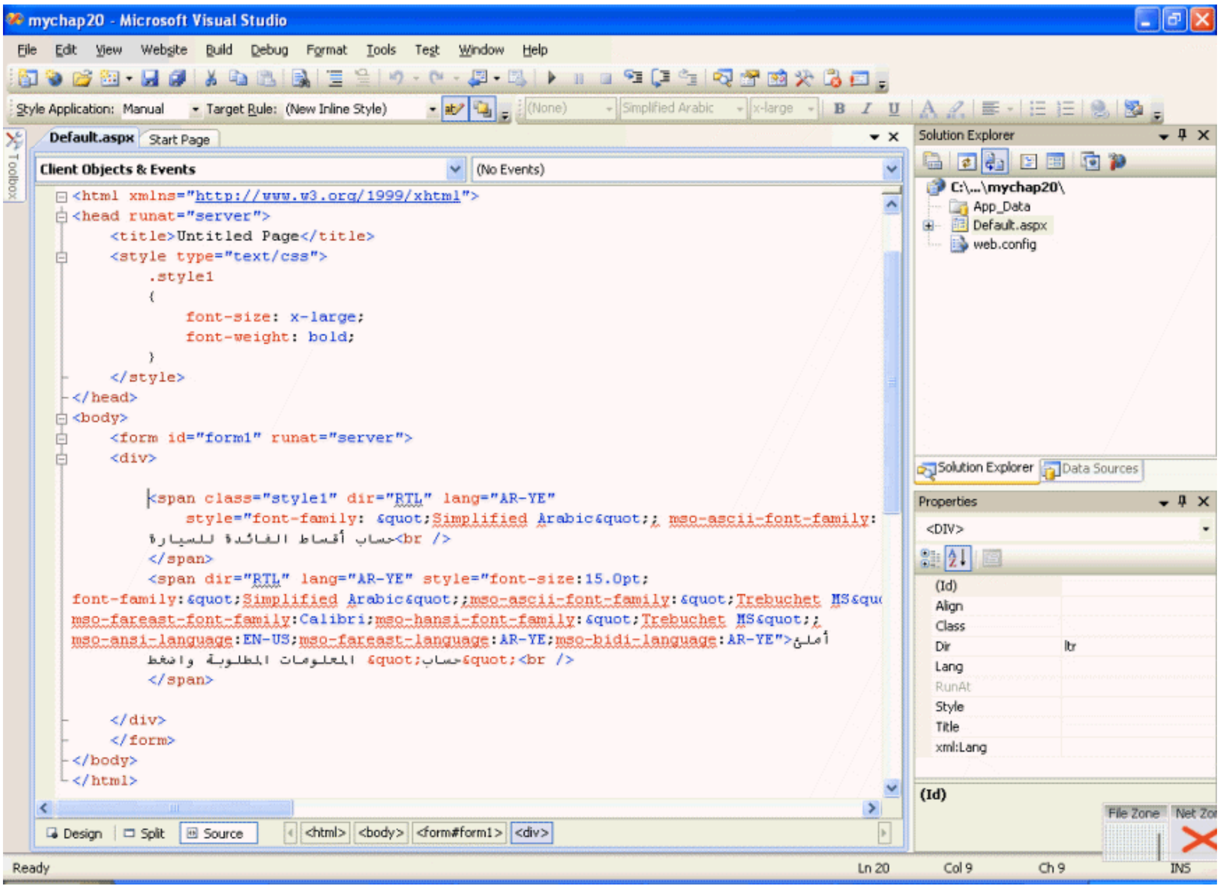
٥- تختلف صفحات الويب عن تطبيقات الويندوز بأنك تستطيع إضافة النصوص إليها مباشرة بدون الحاجة إلى صندوق النص كما في برامج الويندوز، فإذا كنت في الخيار Design تستطيع مباشرة كتابة النص، عند الخيار Source سترى النص الذي قمت بإضافته

موجوداً ولكن وسط أكواد الـــ HTML، في حالة التصميم Design سيظهر النص من الأعلى إلى الأسفل كما هو الحال عند طباعة النصوص في برنامج الوورد Word ولن تظهر أكواد الـــ HTML في مرحلة التصميم، فإذا كنا الآن في الخيار Design سنشاهد مستطيل نقوم بالكتابة فيه (مع العلم بأننا سنقوم بتصميم تطبيق يقوم بحساب الفوائد الربوية على سعر سيارة، نعرف أن الربا حرام لكننا هنا مع التطبيق المُقَدَم من مؤلف الكتاب وسنستفيد من فكرة التطبيق البرمجية مع يقيننا بحرمة الربا) نقوم بالكتابة في المستطيل ما يلي "حساب أقساط الفائدة للسيارة" وتحتها مباشرة نكتب " أملئ المعلومات المطلوبة واضغط "حساب" ".

٦-سنقوم الآن ببعض التغييرات على العنوان الذي كتبناه وعلى النص الذي تحته، إذا لم نشاهد صندوق الأدوات Formatting في بيئة التطوير ننقر بيمين الماوس على الشريط العلوي لبيئة التطوير ثم نختار صندوق الأدوات Formatting من القائمة التي تظهر فسيقوم الفيجوال بإضافته مباشرة، بعد إضافته نظلل العنوان "حساب أقساط الفائدة للسيارة" ثم من شريط Formatting نختار الرمز B وحجم الخط ٢٤، لتكون الشاشة كما في هذه الصورة:



٧-لنذهب الآن نرى كود الــ HTML، فكما وضحنا سابقاً نقوم نحن بعملية التصميم ويقوم الفيجوال بيسك بإضافة كود الــ HTML في التبويب Source، نضغط على التبويب Source وسنرى كود الــ HTML كما في هذه الصورة:



لاحظ كيف قام الفيجوال بيسك بكتابة أمواد الــ HTML، قم بقراءة الكود أكثر من مرة لتعرف كيف قام بكتابة الكود لكل سطر وكيف كتب مقاس الخط، لاحظ كذلك كيف قام الفيجوال بيسك بتقسيم الكود على أساس علامات الــ HTML، فوضع كل كما كتبنا بين الكلمتين <body> و </body> ولاحظ أيضاً أن كل تبويب Tag من تبويبات الــ HTML تبدأ بين القوسين <> وتنتهي بنفس الكلمة ولكن بين العلامتين </>.

٨-نذهب إلى صندوق الأدوات لإضافة بعض الأدوات (يجب العلم بأن أدوات الويب تتشابه مع أدوات تطبيقات الويندوز وهناك بعض الفروق مثلاً لإضافة أداة ويب إلى التطبيق لابد من النقر بالماوس مرتين على الأداة Double-Click من أجل إضافتها للتطبيق الويب)، نذهب إلى صفحة الويب التي نقوم بتصميمها ونذهب إلى علامة إدخال النص (الإشارة)

الصغيرة (العمود الصغير) التي تظهر وتختفي في نهاية النص الذي نقوم بكتابته) ونذهب بها إلى نهاية السطر الثاني الذي كتبناه سابقاً، ثم نضغط Enter على الكيبورد ثلاث مرات لترك مسافة بين النص المكتوب وبين المكونات التي سنضيفها إلى الصفحة لإن المكونات ستُضاف إلى نفس النقطة الموجود فيها إشارة الإدخال.

ملاحظة: يقوم مصمم الويب ضمن بيئة الفيجوال بترتيب الكائنات على صفحة الإنترنت بشكل أوتوماتيكي على أساس العلاقة بين الكائنات، وهذا فرق مهم بين تطبيقات الإنترنت وتطبيقات الويندوز، حيث نستطيع وضع الكائنات في المكان الذي نريد على تطبيقات الويندوز، نستطيع تغيير أماكن الكائنات على صفحة الويب بالشكل الذي نريد باستخدام ما يُسمى بـــ التحريك الكامل absolute positioning لكننا سنلاحظ العديد أكثر من ردة فعل من أكثر من متصفح.

٩- ننقر مرتين Double-click على صندوق النص في صندوق الأدوات ليقوم البرنامج بإضافة صندوق نص إلى مكان إشارة الإدخال. ستلاحظ ظهور النص asp:textbox#TextBox1 فوق صندوق النص الذي قمنا بإضافته حيث ينبه ASP.NET أن هذا الكائن يتبع له، لا تقلق سيختفي هذا النص عند تشغيل التطبيق.

١٠- نضغط خارج صندوق النص لنغير مكان إشارة الإدخال إلى ما خلف صندوق النص ثم نضغط Enter مرتين، ثم نضغط على أداة صندوق النص مرتين لنضيف صندوق النص الثاني لتطبيقنا.

١١- نكرر العملية السابقة لنضيف صندوق نص ثالث للتطبيق، سنقوم الآن بإضافة ليبلات Labels للتطبيق فنذهب بالماوس إلى يمين صندوق النص الأول وننقر لنقل إشارة الإدخال إلى يمين صندوق النص الأول لنقوم بإضافة ليبل، نضغط مسافة مرتين لإنشاء فراغ بين صندوق النص والليبل.

١٢- نضيف الآن ليبل للتطبيق بالنقر عليه مرتين في صندوق الأدوات.

١٣- نكرر العملية السابقة مرتين لإضافة ليبل لصندوق النص الثاني والثالث.

١٤- ننقر على يمين الليبل بجانب صندوق النص الثالث ثم نضغط على Enter مرتين للإنتقال سطرين بإشارة الإدخال.

١٥- ننقر مرتين Double-Click على الزر Button في صندوق الأدوات لإضافة زر إلى نهاية صفحة الإنترنت، ستكون شاشتك كما في هذه الصورة:



١٦- سنذهب الآن إلى نافذة الخصائص لتعديل خصائص الكائنات التي قمنا بإضافتها لصفحة الويب، إذا لم تكن نافذة الخصائص ظاهره نضغط على F4 لتظهر، الآن سنقوم ببعض التعديلات على الخصائص، أول فرق بين تطبيقات الويب وتطبيقات الويندوز ستلاحظه في نافذة الخصائص هو تَغيُر اسم الخاصية Name إلى ID، وبالرغم من هذا التَغَيُر فلا تزال تحتفظ بنفس الأداء للخاصية Name.

١٧- نقوم بتعديل الخصائص للكائنات كما في الجدول التالي:

الكائن	الخاصية	الضبط
TextBox1	ID	txtAmount
TextBox2	ID	txtInterest
TextBox3	ID	txtPayment
Label1	ID	lblAmount
Label1	Text	مبلغ القرض
Label2	ID	lblInterest
Label2	Text	نسبة الفائدة (مثلاً ٩%)
Label3	ID	lblPayment
Label3	Text	القسط الشهري
Button1	ID	btnCalculate
Button1	Text	حساب

ستكون صفحة التصميم كما في هذه الصورة:



كتابة الأكواد للكائنات على صفحة الويب

نفس طريقة كتابة الكود لتطبيقات الويندوز ننقر مرتين على الكائن في صفحة التصميم ثم نقوم بكتابة الكود، بعد إنشاء صفحة الويب يستطيع المستخدم رؤية الكائنات في الصفحة لكنه لا يستطيع مشاهدة الكود، لإن الكود يَتم تنفيذه على خادم الإنترنت السيرفر ولا تظهر للمستخدم إلا النتيجة، وإذا حاول المستخدم فسيظهر له كود التصميم بالـــ HTML فقط بدون الكود الأصلي للصفحة، فعندما يقوم المستخدم بالنقر على زر في صفحة إنترنت يقوم الزر بإرسال النقرة إلى خادم الإنترنت Web Server ويقوم الخادم بتنفيذ الكود بداخل السيرفر ثم إرسال صفحة إنترنت جديدة للمستخدم.

١- سنقوم الآن بكتابة الكود للزر "حساب" وذلك بالنقر مرتين على الزر لتظهر لنا منطقة الكود عند الحدث Click التابع للزر "حساب"، نكتب الكود التالي:

```
Dim LoanPayment As Double
'Use Pmt function to determine payment for 36 month loan
' نستخدم الدالة Pmt
' لتحديد الأقساط لفترة ٣٦ شهر
LoanPayment = Pmt(CDbl(txtInterest.Text) / 12, 36, CDbl(txtAmount.Text))
txtPayment.Text = Format(Abs(LoanPayment), "$0.00")
```

استخدمنا الدالة المالية Pmt لتحديد قيمة الأقساط خلال الفترة ٣٦ شهر بال'تماد على نسبة الفائدة المكتوبة في صندوق النص txtInterest، تتم حفظ قيمة المتغير بداخل المتغير LoanPayment ثم تتم عملية تهيئة Format للمبلغ ليظهر على هيئة مالية، ويعرض في صندوق النص txtPayment، أما الدالة CDbl فتستخدم لتحويل المبلغ المكتوب في صندوق النص txtInterest من النوع String (نصي) إلى النوع Double (رقم حقيقي)، أما الدالة ABS لعرض القيمة الغير النسبية للعدد وتستخدم الدالة لعرض قيمة مبلغ القسط كرقم موجب، يوجد خط أزرق متعرج تحت الدالة ABS وسيذهب هذا الخط المتعرج بعد استيراد مجال الأسماء System.Math والذي سنقوم فيه في الخطوة التالية، لماذا نقوم بتحويل الرقم إلى رقم موجب

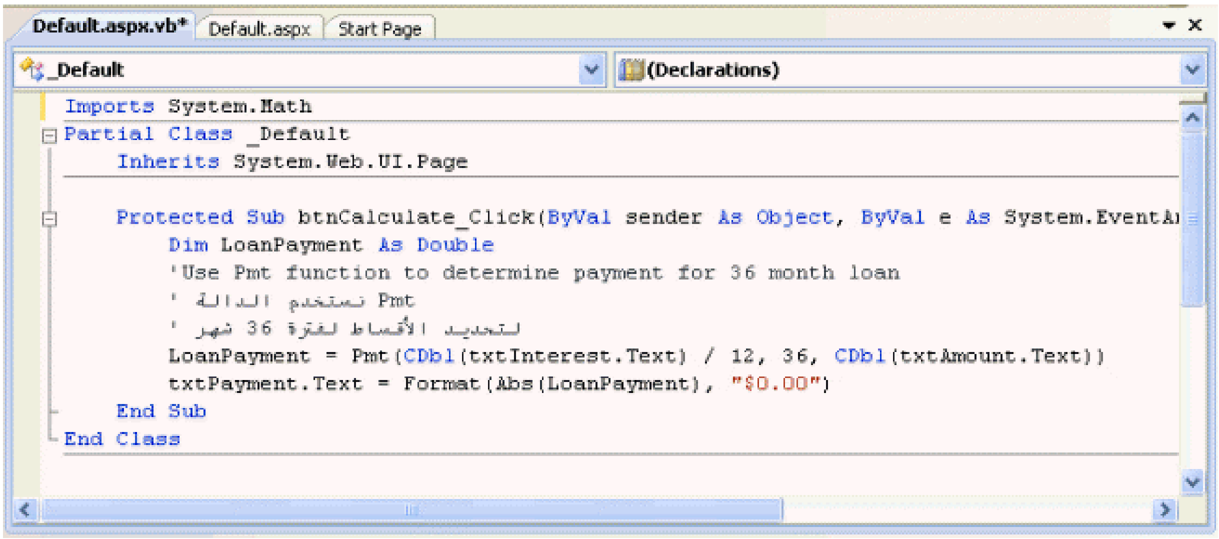
بواسطة الدالة ABS لإن الدالة Pmt ترجع لنا رقم سالب على اعتبار أن الرقم مدين وليس دائن، ولذلك كان لابد من استخدام الدالة ABS لتحويل الرقم إلى رقم موجب.

٢-نذهب إلى أعلى منطقة الكود ونقوم باستيراد مجال الأسماء وذلك بكتابة الكود التالي في أعلى منطقة الكود:

```
Imports System.Math
```

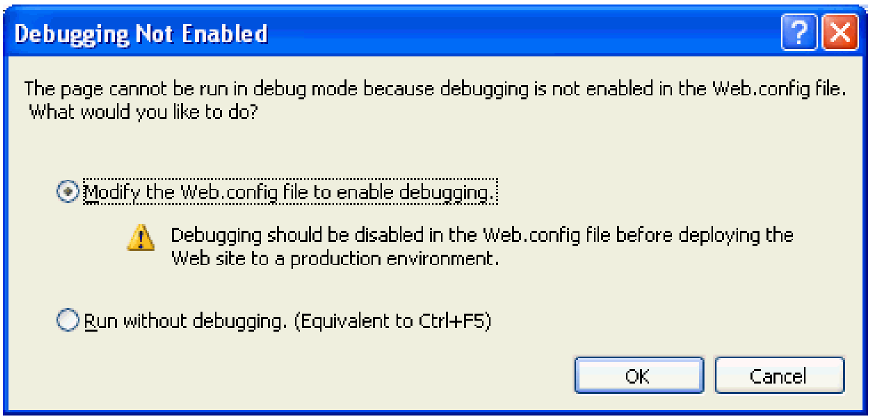
نلاحظ أن الكود المكتوب هو نفس الكود الذي تعاملنا معه خلال البرمجة بالفيجوال بيسك في هذا الكتاب، كما تعلمنا في الفصل الخامس بأن الدالة Abs ليست مضمنه بشكل تلقائي بداخل الفيجوال بيسك، لذلك لابد من استيراد الفئة الخاصة بها وهي Math، بعد عملية الاستيراد سيزول الخط الأزرق المتعرج من تحت اسم الدالة.

سيكون شكل محرر الكود كما في الصورة التالية:



٣-نقوم بحفظ التغييرات.

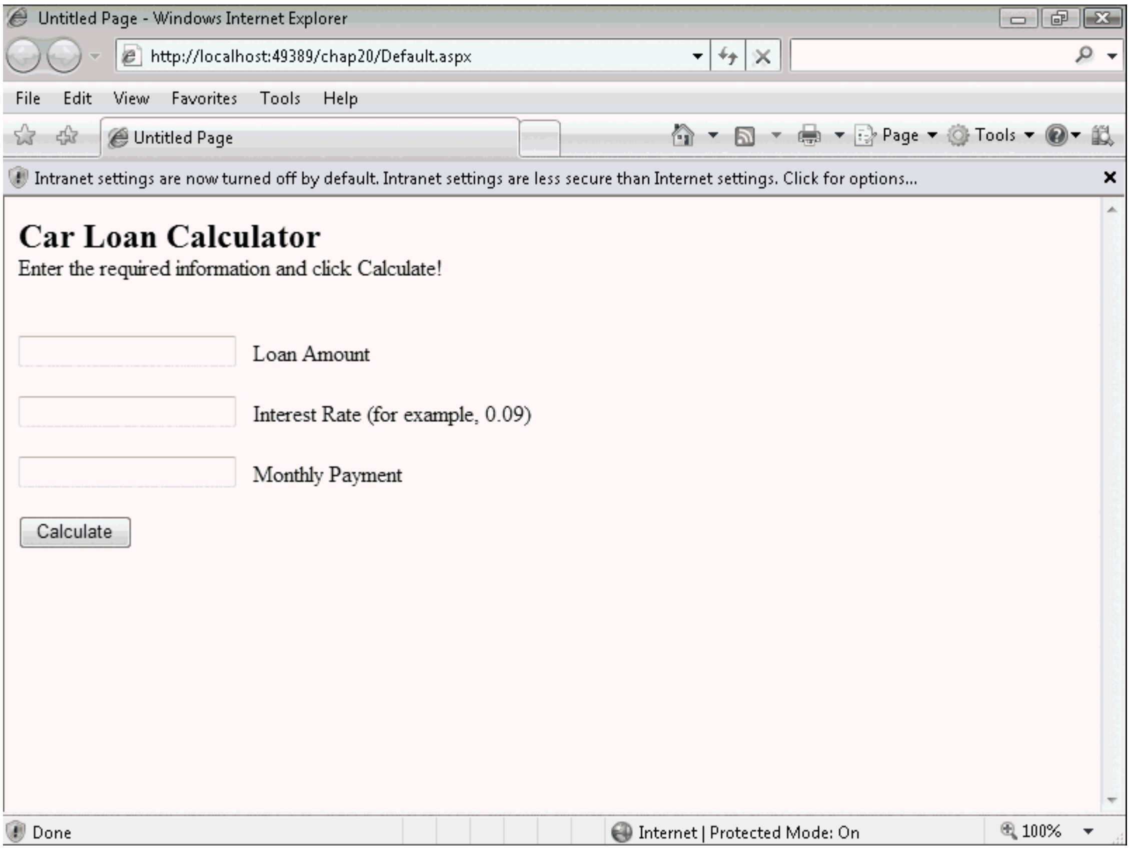
٤-نقوم الآن بتنفيذ التطبيق، لتظهر لنا هذه النافذة:



هذه النافذة تعرض لنا تنبيه أمني بخصوص تعديل الملف Web.Config نستطيع تجاوز هذا التحذير باختيار الخيار الثاني، لكن من الأفضل التعديل على الملف Web.Config الآن، ملاحظة: قبل نشر تطبيق الويب أو صفحة الويب تأكد من تعديل الملف Web.Config لتمنع التعديل عليه من قبل المخربين.

٥-نضغط Ok ليقوم بتعديل الملف.

٦-نقوم بتنفيذ البرنامج بالضغط على F5، ستظهر لنا مثل هذه النافذة بداخل المتصفح الرئيسي في الجهاز:



طبعاً هذه الصورة مأخوذة من الكتاب الأصلي للمؤلف، حيث لم استطع تنفيذ المشروع في جهازي بسبب بعض المشاكل في الـــ ASP.NET المنصب في جهازي، على كل حال كُل من اتبع الخطوات المذكورة سيشتغل عنده التطبيق في متصفح الإنترنت، إذا كان المتصفح Internet Explorer فلدواعي أمنية قد يسألك المتصفح بواسطة شريط في أعلى الصفحة هل تريد تفعيل الصفحة وهل تثق بما فيها فتختار نعم، أما إذا كان المتصفح FireFox ومحمل عليه الإضافة Noscript التي تمنع الأكواد على صفحات الإنترنت من العمل قد تحتاج للسمح لأكواد هذه الصفحة بالعمل.

٧-قم بكتابة مبلغ معين في الخانة المقابلة لمبلغ القرض ولنفرض أنه ١٨٠٠٠، وحدد نسبة الفائدة بـــ 09.% ثم اضغط الزر "حساب"، سيقوم التطبيق مباشرة بحساب القسط الشهري بحسب الدالة المكتوبة بداخل الكود وكتابة الناتج ليكون ٥٧٢,٤٠.

٨-نقوم الآن بإغلاق متصفح الإنترنت، كما تلاحظ بأن إنتاج تطبيقات الإنترنت يشابه إلى حد كبير إنتاج تطبيقات الويندوز، فقط تطبيقات الإنترنت تعمل على متصفحات الإنترنت، تستطيع إنشاء break points لمراقبة تنفيذ الكود في برنامجك كما في تطبيقات الويندوز بالضبط. لنقل الصفحة المصممة إلى خادم الويب ننقل ملف الـــ aspx. والملفات المساعدة (إن وجدت) إلى خادم الويب Server يعمل على ويندوز ويدعم الـــ ASP.NET وسيعمل تطبيقنا مباشرة.

التأكد من مُدخلات المستخدم

في تطبيقنا السابق وعندما يقوم المستخدم بإدخال القيم قد يحدث خطأ معين في حالة نسيان قيمة حقل معين أو غير ذلك، ولتلافي هذه الأخطاء نستفيد من أدوات التأكد الموجودة تحت قائمة Validation في صندوق الأدوات، لتتعلم أكثر عن هذه الأدوات راجع التعليمات المرفقة مع الفيجوال بيسك Documentation.

إضافة صفحات أُخرى ومصادر لتطبيق إنترنت

الآن بدأنا نستفيد من تطبيقات الإنترنت بحيث نستطيع إضافة مكونات جديدة للتطبيق مثل صفحات الـــ HTML و الملفات النصية وملفات الـــ XML، وكذلك قواعد البيانات وخرائط للموقع وخدمات الويب وغيرها من الإضافات، ففي حالة إضافة صفحة HTML مثلاً، نستطيع ذلك بإحدى طريقتين:

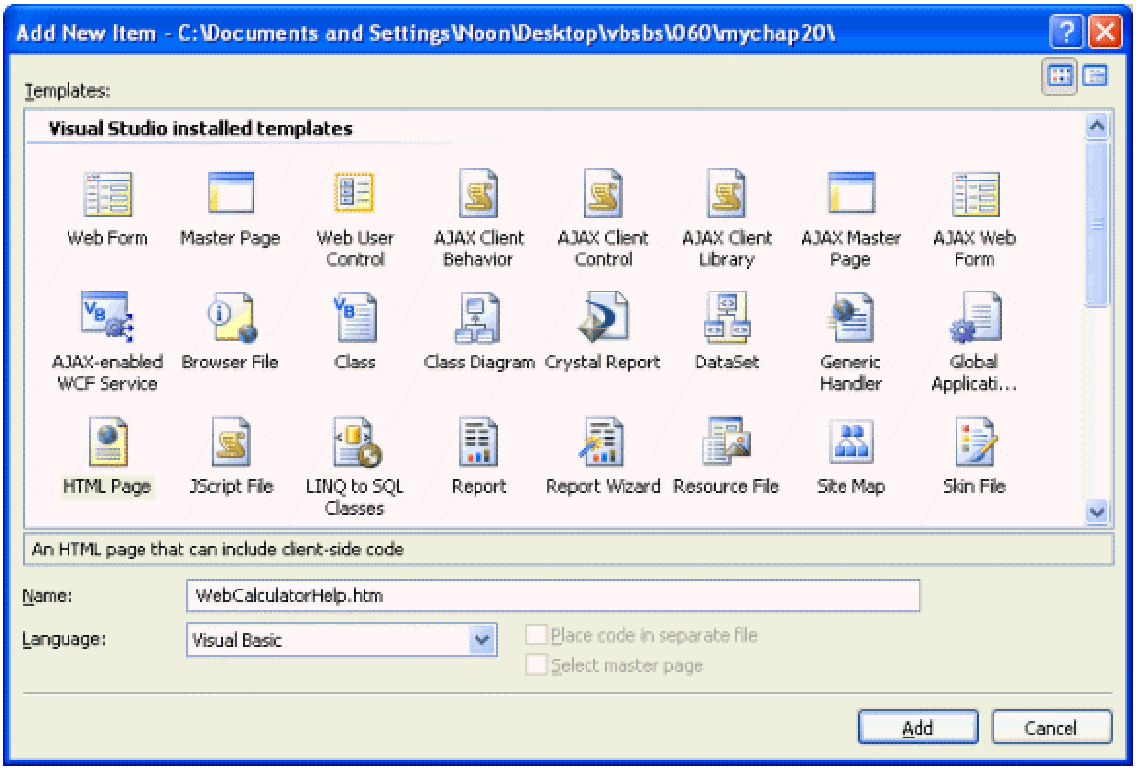
- بواسطة Add New Item من قائمة Website في بيئة التطوير، بعد إضافتها نستطيع إضافة نصوص وكذلك كائنات الـــ HTML في صفحة التصميم.
- بواسطة Add Existing Item من قائمة Website في بيئة التطوير، بحيث نقوم بإضافة صفحة HTML قد تم تصميمها مسبقاً، تستطيع إضافة صفحة أو أكثر.

وللتنقل من صفحة إلى أُخرى في تطبيق الإنترنت نستخدم الكائن HyperLink الذي يقوم بالانتقال من صفحة إلى أُخرى في المتصفح، ونعدل على الخاصية NavigateUrl لتحديد مكان الصفحة الجديدة، سنأخذ مثال على ذلك بالاستفادة من التطبيق السابق حيث سنقوم بإضافة صفحة جديدة

للتطبيق تشرح كيفية التعامل مع التطبيق، وللربط بين الصفحتين سنستخدم الكائن HyperLink لنضعه في الصفحة الأولى ونربطه مع الصفحة الثانية.

١. بالاستفادة من المشروع السابق (موجود نسخة في المرفق ٠٦٠)، من قائمة Website نختار Add New Item، ستفتح لنا نافذة لنختار ما نريد منها لإضافته للمشروع، نختار منها HTMLPage وهي صفحة انترنت بصيغة HTML والتي سنضيف إليها كائنات الـــ HTML فقط ولا نستطيع إضافة كائنات الخوادم server controls لأنها صفحة HTML، والفرق بين كائنات الـــ HTML وكائنات الخوادم server controls هو أن كائنات الـــ HTML تعمل على المتصفح بدون الرجوع إلى خادم الويب Server بعكس كائنات الخوادم server controls.

٢. نسمي الصفحة WebCalculatorHelp.htm قبل إضافتها إلى المشروع، كما في هذه النافذة:



٣. نختار Add، ستضاف الصفحة مباشرة إلى مستكشف المشروع Solution Explorer، وستُفتح في مصمم الصفحات، علي اليسار في صندوق الأدوات ستلاحظ ظهور أدوات الـــ HTML فقط لإن الصفحة صفحة HTML.

٤. نقوم بكتابة هذه النص في الصفحة المُضَافة:

حساب أقساط السيارة

صفحة حساب أقساط السيارة تم تصميمها بواسطة كتاب تعلم فيجوال بيسك ٢٠٠٨ خطوة خطوة، بواسطة المؤلف مايكل هال فرسون، لتتعلم أكثر عن تصميم المواقع باستخدام ASP.NET إقرأ الفصل العشرين من الكتاب المذكور.

تعليمات الاستخدام

قم بكتابة مبلغ القرض بدون كتابة الفواصل أو علامة العملة في خانة مبلغ القرض. قم بكتابة قيمة الفائدة في خانة الفائدة ولابد أن تكون الفائدة بين ٠ و ١، ففي حالة كانت الفائدة ٩%
نكتب ٠٩.

مع ملاحظة أن هذه الحاسبة تحتسب مبلغ القسط باعتماد على الفائدة المكتوبة وفترة تسديد ٣٦ شهر، بعد إكمال البيانات المطلوبة اضغط على الزر "حساب".

٥- نقوم بتعديل النص وإضافة التغييرات ليظهر كما في هذه الصورة:



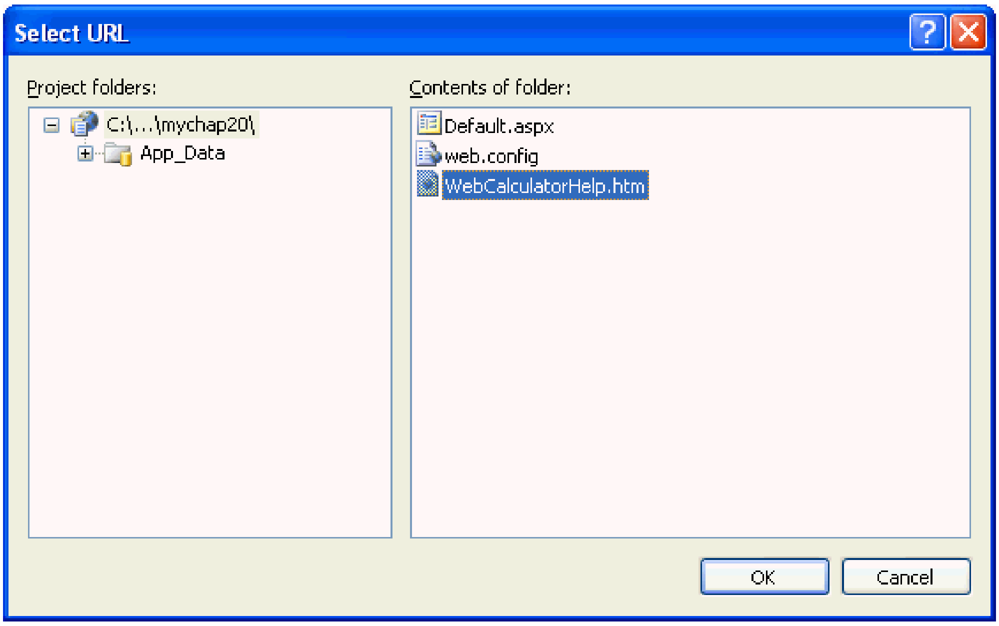
نقوم بحفظ التغيرات ثم نتجه إلى الصفحة الأولى لنقوم بإضافة الكائن HyperLink ليقوم بالربط بين الصفحات.

٦-بعد استعراض الصفحة الأُولى Default.aspx، نتأكد من أن علامة الإدخال في أسفل الصفحة ثم ننقر مرتين على الكائن HyperLink، لتقوم بيئة التطوير بإضافته إلى الصفحة.

٧-نضبط الخاصية Text للكائن HyperLink على "تعليمات"، نضبط كذلك الخاصية ID على lnkHelp.

٨-نذهب إلى الخاصية NavigateUrl ونضغط على الزر المقابل لها ...، تفتح لنا نافذة لاختيار أو لكتابة وصلة الصفحة التي نريد من الصفحة أن تفتحها في حالة النقر على النص "تعليمات"، من النافذة المفتوحة نختار الصفحة WebCalculatorHelp.htm كما

في الصورة:



٩- نضغط Ok لنقوم بعدها بحفظ التغيرات، ثم نقوم بتنفيذ التطبيق، ونقوم بإدخال رقم في "مبلغ القرض" ثم نسبة الفائدة ثم نضغط حساب، ونجرب هذه المرة ننقر فوق "تعليمات" وستفتح لنا صفحة جديدة في المتصفح فيها التعليمات التي قمنا بكتابتها في الصفحة المُضافة الثانية.

المشروع بالمرفق رقم ٠٦٠، تَخيل معي الآن تطبيق انترنت يحتوي على قاعدة بيانات، بالطبع سيكون أكثر ديناميكية واحترافا، تفضل فيما يلي تطبيق عن استخدام قواعد البيانات في تطبيقات الويب.

استعراض سجلات قواعد البيانات على صفحات الإنترنت

على صفحات الإنترنت نجد الكثير الكثير من المعلومات، معظم الأحيان لا نهتم بالكم المعلوماتي بقد ما نهتم بالكيف، فكيف نستعرض المعلومات التي نحتاجها وكيف نصل إلى المعلومة المطلوبة بأيسر الطرق، من أجل هذا أنشأ تم إضافة قواعد البيانات إلى صفحات الإنترنت، فإذا أردنا معرفة تفاصيل شحنة الـــ DHL أو الـــ UPS علينا أن نفتح صفحة شركة الشحن ونكتب رقم الشحنة لمعرفة بقية التفاصيل من المؤكد أن عملية تتبع الشحنة تعتمد على قاعدة بيانات عملاقة تحتفظ بكل التفاصيل عن جميع الشحنات في العالم وتقوم بعرض المعلومات الخاصة بشحنتنا فقط.

بالاستفادة من المشروع السابق في المرفق رقم ٠٦٠ سنقوم بربط المشروع مع قواعد البيانات.

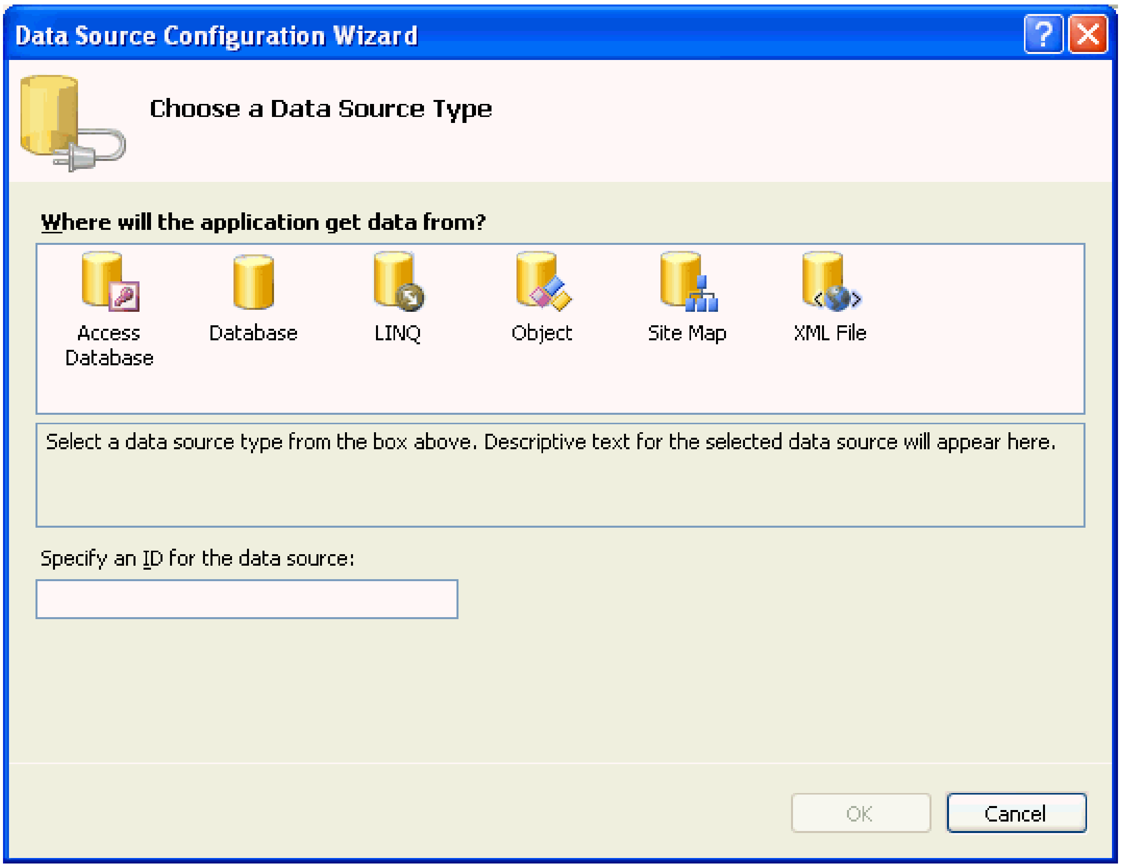
١- من قائمة Website نختار Add New Item.

٢- نختار Web Form ونغير الاسم إلى InstructorLoans.aspx ونضغط على Add.

٣- نقوم بكتابة النص التالي في أعلى الصفحة المُضافة: "الجدول التالي يوضح أسماء المعلمين الذين يحتاجون قروض السيارة بالإضافة إلى أرقام هواتفهم" بعدها نضغط Enter مرتين لنضع فراغاً قبل إضافة الكائن الجديد لصفحة الإنترنت، حيث تقوم بيئة التطوير بإضافة الكائن الجديد في مكان علامة الإدخال.
الآن سنقوم بالتواصل مع قاعدة البيانات Students.mdb مع الجدول Instructors وسنقوم بالربط مع عمودين فقط من الجدول، لنستعرض بيانات العمودين على صفحة انترنت بواسطة المكون GridView والذي يتشابه إلى حد كبير مع المكون DataGridView الذي استخدمناه في الفصل التاسع عشر من هذا الكتاب، لكن مكون الجدول GridView مصمم خصيصاً لتطبيقات الإنترنت، قاعدة البيانات المستخدمة الآن من النوع مايكروسوفت أكسس لكننا نستطيع استخدام قواعد البيانات من النوع SQL.

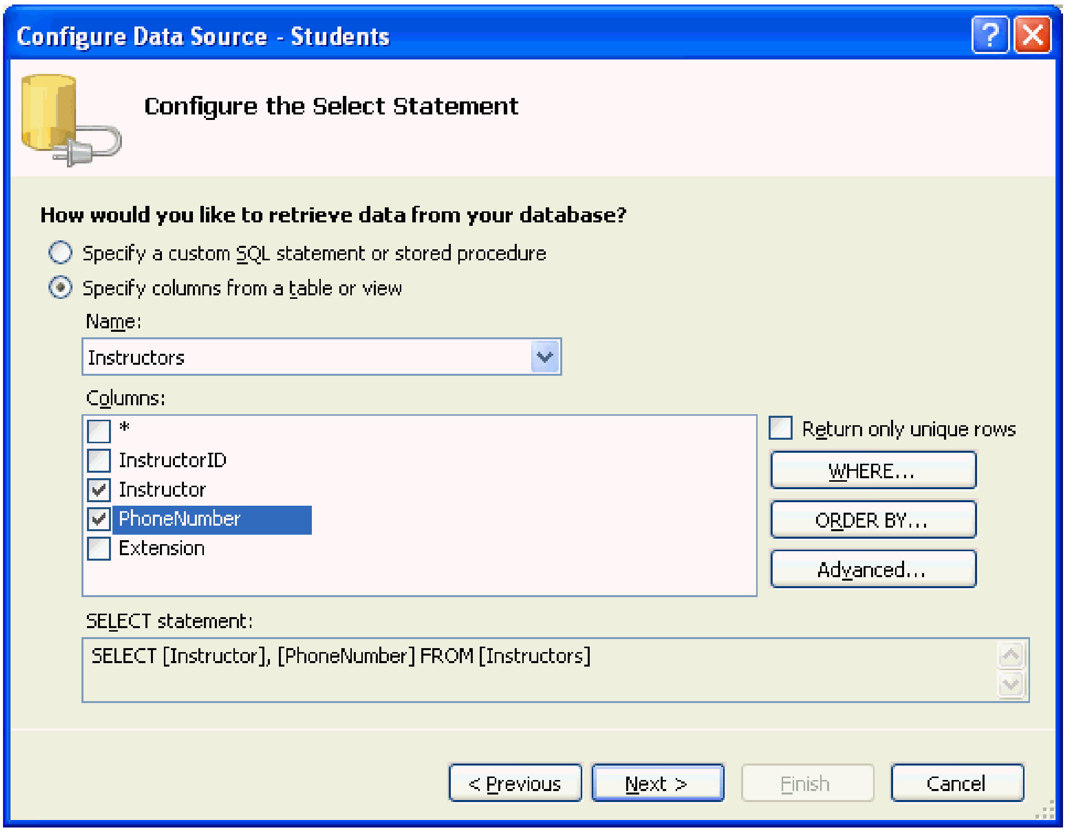
٤- عند وصول علامة الإدخال إلى المكان المناسب ننقر مرتين على المكون GridView في صندوق الأدوات، ستقوم بيئة التطوير بإضافة المكون GridView1 مباشرة إلى الصفحة ونقوم بالضغط على السهم الصغير أعلى يسار المكون ونختار Choose Data Source ومنها نختار <New Data Source>، ستقوم بيئة التطوير بفتح نافذة لاختيار قاعدة

البيانات كما في الفصل التاسع عشر كما في الصورة التالية:



٥- نختار Access Database ونكتب Students في صندوق النص بالأسفل ثم نضغط Ok.

٦- سيطلب منك البرنامج تحديد مسار قاعدة البيانات والتي أرفقتها لك في المرفق رقم ٠٦١ وهي نفس قاعدة البيانات التي استخدمناها في الفصلين الثامن والتاسع عشر، نقوم بتحديد مسار قاعدة البيانات Students.mdb ونختار منها الجدول Instructors، ومن الجدول نختار العمودين Instructor و PhoneNumber كما في الصورة التالية:

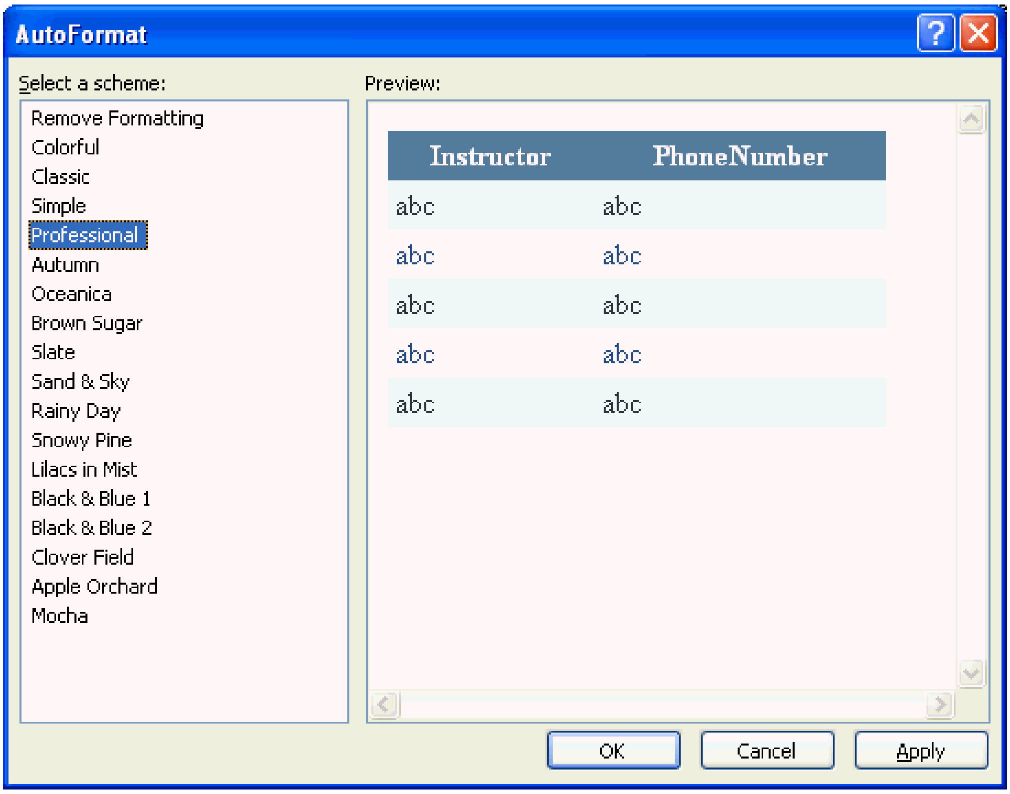


لاحظ أسفل النافذة ستلاحظ بأن البرنامج قام بكتابة جُمل وأوامر الــ SQL على الأعمدة التي اخترتها من قاعدة البيانات.

٧- نختار Next لتجربة جُملة الــ SQL ثم نضغط على الزر Test Query في النافذة التي تليها، سنلاحظ ظُهور بيانات العمودين المُختارين.

٨- نضغط Finish لإغلاق المعالج، سيقوم الفيجوال بيسك بإغلاق المعالج ثم تعديل عدد الأعمدة وعناوين الأعمدة وكذلك كتابة abc في خلايا الأعمدة.

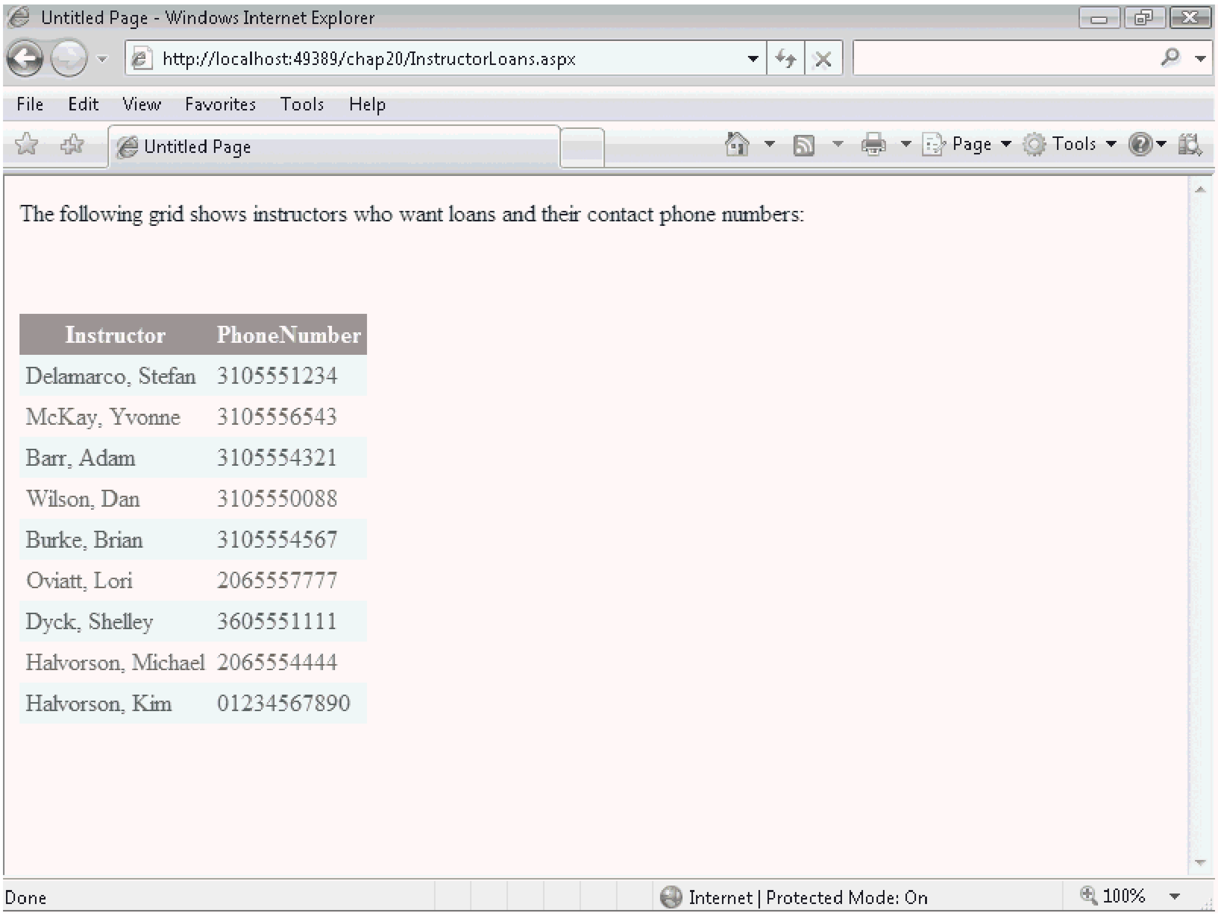
٩-نضغط السهم الصغير مرة ثانية ونختار Auto Format من القائمة المنسدلة ستظهر لنا نافذة نختار منها التنسيق المناسب لعرض الجدول انظر الصورة:



١٠- نختار النموذج Professional ثم Ok.

١١- نذهب للصفحة الرئيسية للموقع لنضيف الكائن HyperLink لنربطها مع هذه الصفحة كما هو الحال مع صفحة التعليمات، نذهب للصفحة Default.aspx ونضع مؤشر الإدخال في نهاية الصفحة وننقر مرتين على الكائن HyperLink لتقوم بيئة التطوير بإضافته للصفحة، نغير الخاصية Text للكائن إلى "قائمة بطالبي القروض" والخاصية ID إلى lnkProspects أما الخاصية NavigateUrl فنفتح الزر المقابل لها لنختار الصفحة InstructorLoans.aspx ونضغط OK. لقد أكملنا هذا التطبيق قم الآن بحفظ التعديلات ثم تنفيذ المشروع لتجربته، أدخل مبلغ القرض ٨٠٠٠ ونسبة الفائدة 08. ثم اضغط الزر "حساب" سيقوم التطبيق بحساب القسط الشهري وإظهار النتيجة ٢٥٠,٦٩$ كمبلغ للقسط الشهري، إذا ضغطنا على الوصلة " قائمة بطالبي القروض" فسيفتح المتصفح صفحة جديدة فيها قائمة مسحوبة من قاعدة البيانات كما في هذه الصورة: (الصورة من

كتاب المؤلف):

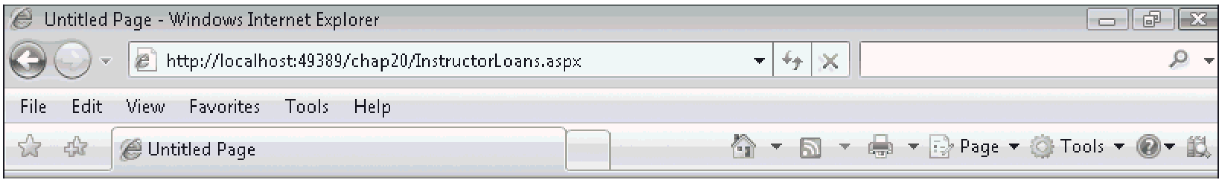


رائع جداً حيث قام تطبيق الإنترنت بعرض محتويات جدول في قاعدة البيانات، وسيحدث هذا في حالة تم رفع تطبيق الإنترنت إلى (خادم الإنترنت) السيرفر الذي يعمل على ويندوز ويدعم الـــ ASP.NET، نستطيع تعديل بعض خيارات الجدول أعلاه لنسمح بترتيب البيانات Sorting وكذلك بتصفح البيانات في صفحات إذا كانت قاعدة البيانات كبيرة، وغيرها من الخيارات.

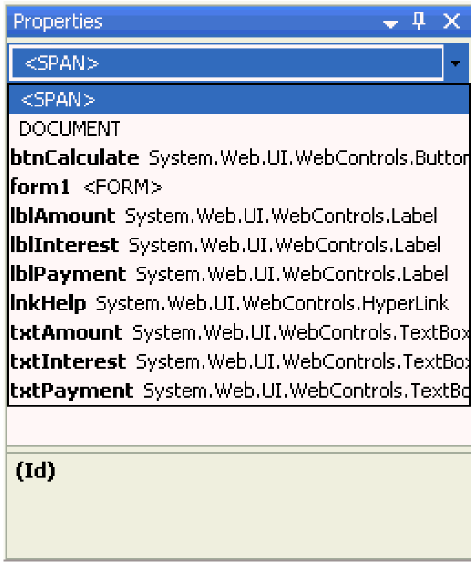
١٢- التمرين موجود في المرفق رقم ٠٦١.

خطوة إضافية للأمام: كتابة عنوان لصفحة الإنترنت ليظهر في المتصفح

قد تكون لاحظت خلال فترة تنفيذ تطبيق الإنترنت الأول الذي قمت بتصميمه بالإعتماد على التمرين المذكور في هذا الكتاب قد تكون لاحظت عدم ظهور عنوان واضح لصفحة الإنترنت كما في هذه الصورة:



فالعنوان في الصورة هو Untitled Page، ولتعديل العنوان نقوم بتعديل الخاصية Title للكائن Document التابع للصفحة، ففي حالة الصفحة Default.aspx وإذا كانت مفتوحة في المصمم نذهب إلى قائمة الخصائص Properties ونفتح القائمة المنسدلة في الخصائص ونختار المكون Document كما في هذه الصورة:



نختار الكائن Document من القائمة ثم نقوم بتعديل الخاصية Title التابعة له إلى " تطبيق حساب أقساط القروض "، بعد تنفيذ التطبيق سيظهر عنوان الصفحة في متصفح الإنترنت بنفس العنوان.

ألف مبروك لقد قمت بإكمال قراءة كتاب "فيجوال بيسك ٢٠٠٨ خطوة خطوة"، أنت الآن جاهز للتحدي الكبير في برمجة تطبيقات الويندوز وتطبيقات الويب لا تتوقف لا تخاف انطلق لميدان التحدي وليس الفتى من قال كان أبي، فهذا الكتاب يضعك في أول درجة في سُلَّم الفيجوال بيسك

ولابد من مشاريع تطبيقيه لتتعلم أكثر، بعد خلاصة الفصل الأخير ستجد عناوين لبعض مواقع الإنترنت إذا أردت تنمية قدراتك في برمجة الفيجوال بيسك واعلم بأنك إذا قرأت كل مراجع الكرة الأرضية في الفيجوال بيسك بدون تطبيق وعمل وتنفيذ للأفكار البرمجية فلن تستفيد ولن تتغير وكأنك يا زيد ما غزيت، فلابد للمتعلم من تطبيق لما تعلمه فابدأ الآن في تصميم برامجك بالفيجوال بيسك وقم بتصميم برامج حسب الطلب لأصدقائك لتتعلم أكثر وتستفيد. يمكنك البداية بمساعدة الآخرين في منتدى فيجوال بيسك للعرب والفريق العربي للبرمجة ثم الانتقال إلى تصميم البرامج حسب الطلب ولو مجاناً أو بمبالغ زهيدة ثم ستكتشف نفسك أو بالأصح سيكتشفك المجتمع بأنك مبرمج موهوب ومطور محترف من الدرجة الأُولى، حينها لا تنتظر قم بنشر برامجك وتطبيقاتك وقم بتسويقها و...و والذي هو ليس موضوع الكتاب.

خلاصة الفصل الأخير

من اجل أن	**قم بالتالي**
إنشاء موقع بـــ ASP	من قائمة File نختار New Web Site ومن النافذة التي تظهر نختار ASP.NET Web Site ثم نختار مسار الصفحة ونختار OK.
التنقل بين التصميم وأكواد التصميم للصفحة	تحت صفحة التصميم نختار Source لعرض الكود أو Design لعرض التصميم.
كتابة نص في صفحة الإنترنت	في صفحة التصميم Design نقوم بكتابة النص مباشرة.
تعديل النص	في صفحة التصميم Design نظلل النص المراد تعديله ثم نقوم بتكبيره أو تصغيره أو إمالة النص أو غير ذلك من التعديلات.
مشاهده أكواد الـــ HTML للصفحة	نختار Source لعرض الكود، والذي يظهر لنا كود الـــ HTML.

إضافة كائنات للصفحة	نذهب بمؤشر الإدخال (وهو المؤشر الذي يظهر في عند نهاية الكتابة) إلى المكان الذي نريد إضافة الكائن ثم النقر مرتين على الكائن.
تغيير اسم الكائن	من الخاصية ID.
كتابة كود للحدث الطبيعي لأي كائن	ننقر مرتين على الكائن في منطقة التصميم ونكتب الكود.
التأكد من البيانات المُدخَلَة من المستخدم	في صندوق الأدوات وتحت القائمة Validation يوجد العديد من الكائنات التي تساعدنا في فحص مُدخلات المستخدم.
تنفيذ تطبيق الإنترنت	بالضغط على F5 كما هو الحال في تطبيقات الويندوز، وستقوم بيئة التطوير بفتح الصفحة في متصفح الإنترنت.
إضافة صفحة HTML إلى تطبيق الويب	من قائمة Web site نختار Add New Item ثم نختار HTML Page ثم نقوم بتصميم الصفحة بالشكل المطلوب.
إنشاء وصلة إلى الصفحات الأُخرى على الصفحة الرئيسية	نضيف الكائن HyperLink إلى الصفحة الرئيسية ثم نعدل الخاصية NavigateUrl لنربط الكائن بالصفحة التي نريد.
إضافة حقول من قاعدة بيانات على صفحة ويب	نضيف المكون GridView إلى الصفحة، ثم نقوم بالربط مع قاعدة البيانات من خلال خيارات الكائن GridView من خلال السهر الصغير أعلى يسار الكائن GridView ثم ربط الكائن بالحقول المرادة من قاعدة البيانات.
تعديل عنوان صفحة الإنترنت	عند تصميم الصفحة نذهب إلى نافذة الخصائص Properties ونختار المكون Document ونعدل الخاصية Title له على العنوان الذي نريد.

إلى أين تذهب لتتعلم أكثر عن الفيجوال بيسك ٢٠٠٨

بحسب المؤلف لقد غطى هذا الكتاب العديد من المراحل التعليمية للفيجوال بيسك ٢٠٠٨ مبتدئ متوسط أو متقدم، لقد زرع في نفسك الثقة بأنك مبرمج تطبيقات الويندوز، لقد تعلمت العديد من التقنيات البرمجية الخاصة بالبرمجة بشكل عام وبفيجوال بيسك بشكل خاص، إذا أردت الاحتراف أكثر في الفيجوال بيسك والبرمجة فاستفد هذه المواقع:

مواقع خاصة بالفيجوال بيسك

بالنسبة للمواقع العربية المتخصصة فيمكنك البدء بالموقعين المشهورين ١- فيجوال بيسك للعرب http://www.vb4arab.com و ٢- الفريق العربي للبرمجة http://www.arabteam2000.com، هناك العديد من المواقع العربية الأُخرى لكن الموقعين أعلاه هما نجوم السماء العربية في تعليم البرمجة.

إذا أردت البحث عبر مواقع البحث فاستخدم المصطلحات: Visual Basic 9 أو Visual Basic 2008 أو Orcas. أما المواقع المتخصصة فمنها:

msdn2.microsoft.com/en-us/vbasic/

وهو موقع مقدم من مايكروسوفت لتعليم الفيجوال بيسك، ويعتبر أفضل موقع على الإطلاق فيما يتعلق بلغة البرمجة فيجوال بيسك، يعرض هذا الموقع تعليمات وأخبار عن لغة البرمجة وكذلك تحديثات ودعم وغيره.

www.devx.com

موقع تجاري يتعلق ببرمجة تطبيقات الويندوز وأيضاً لغة الفيجوال بيسك، أيضا فيه العديد من المقالات والمقارنات بيت لغات البرمجة المختلفة مثل الجافا والدوت نت.

www.microsoft.com/learning/books/

هذه الصفحة تعرض لنا أحدث الكتب التعليمية في لغات البرمجة التابعة لمايكروسوفت.

www.microsoft.com/learning/training/

تعليمي تابع لمايكروسوفت.

هناك أيضاً الموقعين المميَزين ولكنهما باللغة الإنجليزية:

http://www.planet-source-code.com/

http://www.codeproject.com/

بالتوفيق للجميع.

إذا كان لديك أية ملاحظة فالرجاء إرسالها على البريد vbdotnet2008sbs@gmail.com

مروان المفلحي – صنعاء

المدونة البرمجية: http://marwanvb.blogspot.com/